VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Multiple sequence alignment pomocí dynamického programování

Multiple Sequence Alignment by Dynamic Programming

2016 Bc. Martin Rusek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Rusek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Multiple sequence alignment pomocí dynamického programování
Multiple Sequence Alignment by Dynamic Programming

Jazyk vypracování: čeština

Zásady pro vypracování:

V současnosti existuje celá řada algoritmů pro porovnávání sekvencí. Sekvence můžeme najít v oblastech výzkumu a vývoje, jakými jsou např. analýza DNA, RNA nebo proteiny. Tyto oblasti však nejsou jediné, kde lze tyto algoritmy použít. Cílem práce bude prostudovat vybrané metody z oblasti porovnávání sekvencí se zaměřením na vhodnost jejich použití v oblasti dopravy, webových logů, business procesů apod. V práci se diplomant zaměří na algoritmy pro porovnávání více než dvou sekvencí najednou.

1. Prostudování vybraných metod pro porovnání sekvencí.
2. Výběr nejvhodnější metody pro porovnání více sekvencí.
3. Implementace vybrané metody.
4. Provedení experimentů nad reálnou datovou kolekcí a jejich vyhodnocení.

Seznam doporučené odborné literatury:

[1] M. Crochemore, Ch. Hancart, T. Lecroq: Algorithms on Strings, Cambridge University Press, 2006 http://www-igm.univ-mlv.fr/~mac/CHL/chl-1-58.pdf
[2] D. J. Russell: Multiple Sequence Alignment Methods. Methods in Molecular Biology, Vol. 1079, 2014. Doi: 10.1007/978-1-62703-646-7
[3] D. Gusfield: Algorithms on Strings, Trees and Sequence. Computer Science and Computational Biology, Cambridge University Press; 1st edition, 1997

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Kateřina Slaninová, Ph.D.**

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 25. dubna 2016

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 25. dubna 2016

Rád bych na tomto místě poděkoval paní Ing. Kateřině Slaninové, Ph.D. a panu Ing. Janu Martinoviči, Ph.D. za jejich rady, protože by bez nich by tato práce nevznikla.

Abstrakt

Zarovnávání sekvencí je oblast analýzy dat, která se často využívá k určení podobnosti vybraných dat a k jejich klasifikaci. Tato práce se tedy zabývá problematikou zarovnávání sekvencí s důrazem na metody využívající dynamické programování, a které umožňují zarovnávat více sekvencí najednou. Konkrétně to jsou metody Dynamic Time Warping, Longest Common Subsequence a jejich vícenásobné varianty. Jednotlivé metody byly implementovány jako DLL knihovna, která následně hostitelské aplikaci umožní pracovat s danými metodami.

Klíčová slova: Zarovnávání sekvencí, vícenásobné zarovnávání sekvencí, Dynamic Time Warping, Longest Common Subsequence

Abstract

Alignment of sequences is an area of data analysis, which is often used to determine the similarity of the selected data and their classification. This work therefore deals with the issue of sequence alignment emphasizing methods which use dynamic programming, and which allow simultaneous multiple sequence alignment. Specifically, these methods are Dynamic Time Warping, Longest Common Subsequence and their multidimensional variants. Individual methods were implemented as a DLL library, which will then allow the host application to work with these methods.

Key Words: Sequence Alignment, Multiple Sequence Alignment, Dynamic Time Warping, Longest Common Subsequence

Obsah

Seznam použitých zkratek a symbolů

SA	–	Sequence Alignment
MSA	–	Multiple Sequence Alignment
DA	–	Diagonal Matrix
GT	–	Guide Tree
GUI	–	Graphic user interface
LCS	–	Longest Common Substring
LCSS	–	Longest Common Subsequence
PLCSS	–	Progressive Longest Common Subsequence
MLCSS	–	Multidimensional Longest Common Subsequence
PLCS	–	Progressive Longest Common Substring
DTW	–	Dynamic Time Warping
MDTW	–	Multidimensional Dynamic Time Warping
MMDTW	–	Multiple Multidimensional Dynamic Time Warping
PDTW	–	Progressive Dynamic Time Warping
MPDTW	–	Modular Progressive Dynamic Time Warping
DLL	–	Dynamic Link Library

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Analýza dat je v dnešní době, ve které jsou informace prakticky uchovávány jen v elektronické podobě, jedním z nejžádanějších nástrojů pro zpracování dat. Zejména teď, kdy objem dat potřebný analyzovat roste závratným tempem. Jednou z oblastí, která do ní patří je i sequence alignment neboli zarovnávání sekvencí, které slouží k určování vzdálenosti, respektive míry podobnosti mezi sekvencemi. Cílem této práce je prostudování vybraných metod z oblasti porovnávání sekvencí se zaměřením na vhodnost jejich použití v oblasti dopravy, webových logů, business procesů apod. Tato práce se tedy zabývá studiem algoritmů pro porovnávání více než dvou sekvencí najednou a jejich následnou implementací. Tato práce je rozdělena na několik ucelených částí.

Obsah této práce začíná druhou kapitolou, která stručně vysvětluje nutnost analýzy dat v moderní době, historii zarovnávání sekvencí a k čemu je zarovnávání sekvencí využíváno. Dále rozebírá rozdíly mezi jednotlivými typy zarovnání, s kterými se můžeme setkat.

V třetí kapitole je podrobně vysvětlena metoda Dynamic Time Warping (DTW) a několik jejích variant zaměřených na vícenásobné zarovnávání. Konkrétně to jsou metody Multidimensional Dynamic Time Warping (MDTW), Progressive Dynamic Time Warping (PDTW), Multiple Multidimensional Dynamic Time Warping(MMDTW) a Modular Dynamic Time Warping (MPDTW).

Ve čtvrté kapitole je probrána metoda Longest Common Subsequence (LCSS) a stejně jako u metody DTW její vícenásobné varianty. Jsou to metody Multidimensional Longest Common Subsequence (MLCSS) a Progressive Longest Common Subsequence (PLCSS).

Předposlední kapitola rozebírá přístup k samotné implementaci daných metod a strukturu aplikace. Také je zde podrobně popsáno grafické uživatelské rozhraní aplikace spolu s jejím ovládáním a strukturou dat, s kterými je aplikace schopna pracovat.

Na závěr jsou porovnány metody DTW a LCSS, spolu s možnostmi dalšího vývoje daných metod, respektive aplikace implementované v rámci této práce.

2 Zarovnávání sekvencí

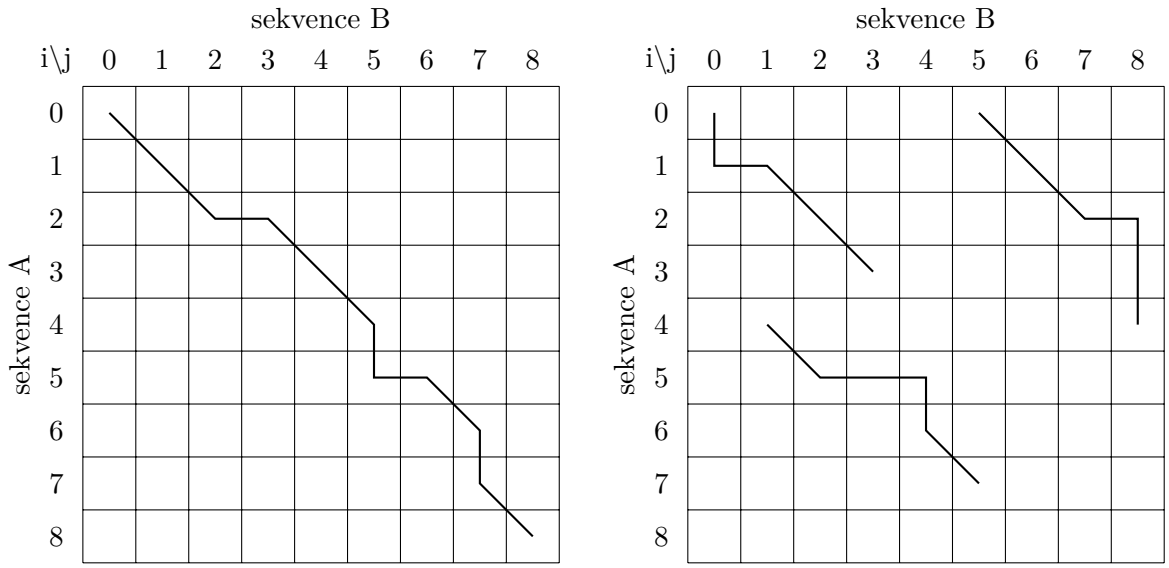
Oblast analýzy dat je v dnešní době elektronických hraček a internetu jednou z nejdůležitějších informačních disciplín, a to zejména v posledních letech, kdy množství dat, které je potřeba analyzovat roste extrémním tempem. I když počítáme s čím dál tím výkonnějším hardwarem, tak v porovnání s tempem jakým roste objem dat je tento nárůst výkonu prakticky zanedbatelný. Díky tomu, je spousta pozornosti věnována studiu a vývoji metod pro analýzu dat. Jedna z oblastí, která do analýzy dat patří je tzv. zarovnávání sekvencí (sequence alignment).

Samotný termín zarovnávání sekvencí je znám již několik desetiletí, ale stejně jako analýze dat je mu věnována nová pozornost. Zarovnávání sekvencí je využíváno v mnoha oborech a pro mnoho disciplín. Například jako je rozpoznávání řeči, kontrola pravosti podpisů, hledání shody otisků prstů, rozpoznávání pohybů lidského těla či hledání podobností mezi různými biologickými sekvencemi, atp..

Existuje mnoho algoritmů pro zarovnávání sekvencí, lišících se svými vlastnostmi. Například metody, které umí pracovat jen s číselnými či kategoriálními sekvencemi, algoritmy pro časové řady, které umí pracovat s dilatací času, dále algoritmy pro biologické sekvence na vyhledání editační vzdálenosti, algoritmy pro analýzu párů sekvencí či celých skupin. Existují specializované metody pro konkretní typy dat, tak i metody obecné, které zarovnání sekvencí generalizují.

2.1 Globální a lokální zarovnání

Zarovnávaní sekvencí se dá dělit na dva základní typy a to na globální a lokální zarovnávání [9] [13]. Globální zarovnání analyzuje vždy celé sekvence. Přesněji řečeno sekvence jsou analyzovány v celku, tedy jedna sekvence je reflektována vůči celé délce sekvence druhé a naopak. Oproti tomu lokální zarovnávání vyhledává shody lokálně, tedy je vhodná pro data, která jsou si částečně či v úsecích podobná. Pro globální zarovnání se používají algoritmy odvozené od Needleman-Wunsch algoritmu a obdobně pro lokální zarovnání od Smith-Waterman algoritmu. Oba tyto algoritmy jsou obecnou předlohou pro metody, které jsou určené pro zarovnávání sekvencí s využitím dynamického programování. Množství metod si tyto algoritmy upravuje pro své vlastní specifické potřeby.



(a) Globální zarovnání (b) Lokální zarovnání

Obrázek 1: Znázornění rozdílu mezi globálním a lokálním zarovnáním

Na obrázku 1 je vidět porovnání mezi globálním a lokálním zarovnáním. Globální zarovnání se snaží mapovat každý element v sekvenci s tím, že první a poslední prvky v sekvencích musí být namapovány na sebe. I když se jedná o globální zarovnání, tak v závislosti na zvolené metodě se i v globálním zarovnání mohou objevit mezery (gaps). Například metoda LCSS vynechává některé body v sekvencích, pokud negativně ovlivňují dané zarovnání. Hodí se zejména na sekvence, které vykazují relativně velkou podobnost a které jsou přibližně stejně dlouhé. Lokální zarovnání na druhou stranu umožňuje vyhledat podobné úseky v sekvencích. Je to užitečné, pokud analyzované sekvence obsahují části, které jsou si podobné, tak i úseky které jsou různé.

Největší rozdíl mezi těmito dvěma algoritmy je ve způsobu hledání zarovnání v matici. U globálního zarovnání se vždy začíná na konci matice (L_A, L_B) 1a, kdežto u lokálního se prohledá celá matice a zarovnání je hledáno od všech maximální či minimálních (záleží na zvolené metodě) pozic.

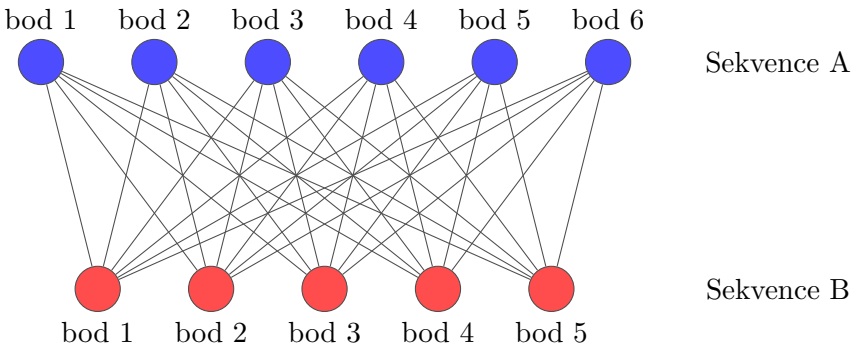
Kromě globálního a lokálního zarovnání se ještě dá setkat s třetím pojmem tzv. semiglobálním (někdy nazývané glocal) zarovnáním, které se využívá, když je požadavek aby výsledné zarovnání obsahovalo začátek a konec analyzovaných sekvencí a zároveň umožňovalo chování lokálního zarovnání.

3 Dynamic Time Warping

Dynamic Time Warping (DTW) je Metoda pro analýzu a mapování sekvencí závislých v čase. Umožňuje porovnat dvojici sekvencí a určit jejich optimální zarovnání [3] [4]. Zarovnáním rozumějme cenu, kolik by stálo převést jednu sekvenci na druhou, respektive určuje vzdálenost (edit distance) mezi dvěma sekvencemi v prostoru. Existují také pokročilejší varianty metody DTW jako je Multidimensional DTW (MDTW), která umožňuje porovnávat sekvence, kde jejich jednotlivé body jsou n-dimenzionální. Například různé křivky, zvukové stopy a podobně. Dále jsou tu metody Multiple Multidimensional DTW (MMDTW) a Progressive Dynamic Time Warping (PDTW), které umožňují zarovnávat více než dvě sekvence najednou. Nakonec je tu metoda Modular Dynamic Time Warping (MPDTW), která kombinuje přístup MMDTW a PDTW metod. Kromě zmíněných metod existuje i spousta dalších variant, které se zaměřují buď na úpravu DTW metod na speciální typy problémů z různých technických oborů, či případně na zefektivnění implementace za účelem optimalizace časové či paměťové náročnosti.

V následující části textu budou podrobně vysvětleny principy všech zmíněných metod, spolu s jejich výhodami a nevýhodami a postup při jejich implementaci.

3.1 Dynamic Time Warping



Obrázek 2: Porovnávání každeho bodu sekvence A s každým bodem sekvence B

Základní myšlenka metody DTW pro určení nejlepšího zarovnání leží v porovnání každého bodu Sekvence A, s každým bodem sekvence B. Nástin tohoto principu lze vidět na obrázku 2. Výpočet DTW metody se skládá ze tří základních, po sobě jdoucích kroků:

1. Vytvoření a inicializace matice vzdáleností (distance matrix) a její následné naplnění.
2. Vyhledání cesty zarovnání (warping path), respektive určení optimálního zarovnání daných sekvencí.
3. Analýza výsledků, určení podobnosti sekvencí a jejich vizualizace.

Definice 3.1 *Dvourozměrná matice vzdáleností (distance matrix) M je A×B, kde* $A = (A_1, A_2, ..., A_i)$ *a* $B = (B_1, B_2, ..., B_j)$ *jsou sekvence.*

Na začátku metody DTW se vytvoří dvojrozměrná matice vzdáleností o rozměrech délek sekvencí, nad kterými provádíme metodu DTW. Mějme sekvence A a B, kde jejich délky jsou L_A a L_B, velikost dané matice bude tedy $L_A \cdot L_B$. Jednotlivé pozice matice jsou vypočteny pomocí následujících vzorců s tím, že první sloupec a řádek jsou inicializovány na ∞ (některé zdroje uvádějí inicializaci na $(i, 0) = i$ a $(0, j) = j$) a počáteční buňka $(0, 0)$ pole je nastavena na 0.

$$D(i,j) = d(i,j) + min \begin{cases} D(i-1,j) \\ D(i,j-1) \\ D(i-1,j-1) \end{cases} \tag{3.1}$$

$$d(i,j) = \mid A_i - B_j \mid \tag{3.2}$$

Kde $D(i, j)$ je buňka v dvojrozměrné matici vzdáleností (distance matrix), i a j jsou souřadnice v matice D, d je vzdálenost (distance) mezi dvěma určitými body v sekvencích na pozicích i a j. A_i a B_j jsou hodnoty v sekvencích.

j		0	1	2	3	4	5	6
i		B	2	7	5	2	4	4
0	A	0	0	0	0	0	0	0
1	0	0	↖ 2	9	14	16	20	24
2	7	0	7	↖ 2	4	9	12	15
3	6	0	11	3	↖ 3	← 7	← 9	11
4	6	0	15	4	4	7	9	← 11

(a) Matice vzdáleností

$$D(3,4) = d(3,4) + min \begin{cases} D(3-1,4) \\ D(3,4-1) \\ D(3-1,4-1) \end{cases}$$

$$D(3,4) = \mid 6 - 2 \mid + min \begin{cases} 3 \\ 9 \\ 4 \end{cases} = 7$$

(b) Výpočet vzdálenosti pro D(3,4)

Obrázek 3: Ukázka výpočtu matice vzdáleností pro metodu DTW

Konstrukce matice vzdáleností probíhá od první buňky $(0, 0)$, po poslední buňku (buňka se souřadnicemi $(L_A - 1, L_B - 1)$) postupně. Metoda DTW a i spousta jiných metod zabývajících se jak zarovnáváním sekvencí, tak i jinými algoritmickými problémy využívá dynamické programování, které slouží k zrychlení výpočtu algoritmů využíváním hodnot z předchozích kroků, které již nemusíme opakovaně počítat znovu. Díky tomu výpočet jednotlivých pozic v matici nesmí být náhodný, je vždy prováděn po řádcích či sloupcích. Důvod je ten, že jednotlivé kroky

algoritmu závisí na výpočtech získaných v předešlých krocích. Jasný příklad tohoto můžeme vidět na obrázku 3. Je zde vidět, že k výpočtu hodnot v matici jsou potřeba sousední prvky směrem k počáteční buňce.

3.2 Nalezení nejlevnější cesty

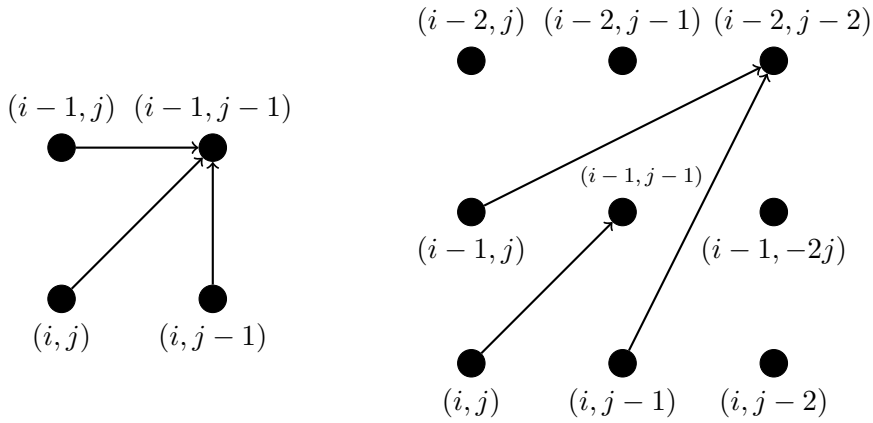
Po naplnění matice vzdáleností je v ní potřeba najít nejkratší cestu (warping path) spojující její první a poslední prvek, respektive pozice v matici se souřadnicemi $(0, 0)$ až $(L_A - 1, L_B - 1)$. Tedy cesta je definovaná jako [4]:

Definice 3.2 *Cesta (warping path) pro dvourozměrnou matici vzdáleností M je sekvence $P = (p_1, p_2, ..., p_L)$, kde $p_l = (A_i, B_j) \in A \times B$ a $l \in \langle 1, L \rangle$ a zároveň splňující tyto podmínky.*

1. Podmínka omezenosti: $p_1 = (1, 1)$ a $p_L = (L_A, L_B)$
2. Podmínka Monotonie: $A_1 \leq A_2 \leq ... \leq A_{L_A}$ a $B_1 \leq B_2 \leq ... \leq B_{L_B}$
3. Podmínka velikosti kroku: $p_{l+1} = p_l \in \{(1, 0), (0, 1), (1, 1)\}$ pro $l \in \langle 1, L - 1 \rangle$

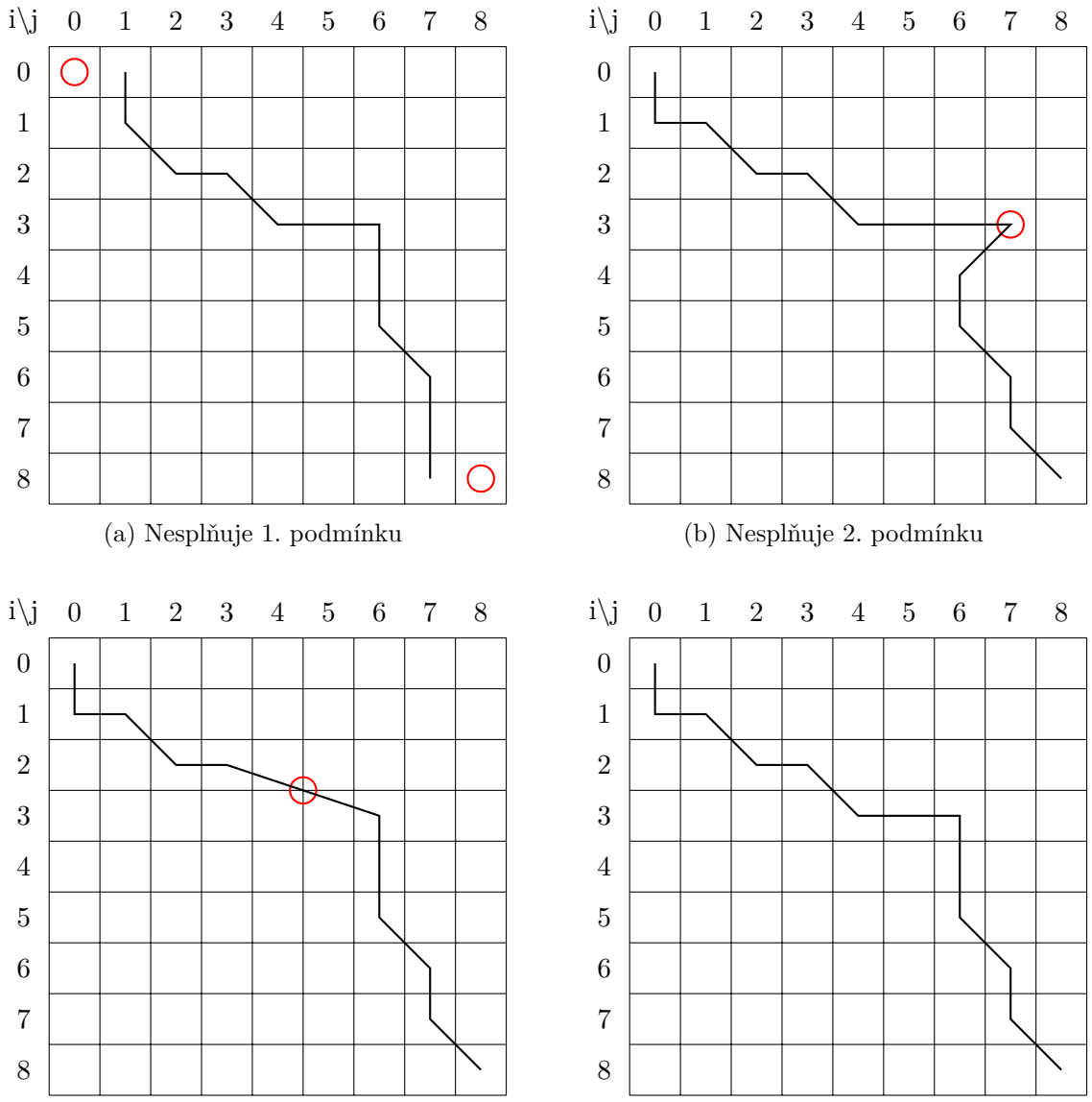
Kde L_A a L_B jsou délky zarovnávaných sekvencí, A_i a B_j jsou body v sekvencích A, B.

Třetí podmínka v některých variantách DTW metody není nebo může být formulována jiným způsobem. Například $p_{l+1} = p_l \in \{(1, 1), (1, 2), (2, 1)\}$ pro $l \in \langle 1, L - 1 \rangle$. Znázornění lze vidět na obrázku 4.



(a) Standardní krok (b) Větší krok

Obrázek 4: Příklady různých velikostí kroků v cestě



(a) Nesplňuje 1. podmínku

(b) Nesplňuje 2. podmínku

(c) Nesplňuje 3. podmínku

(d) Splňuje všechny podmínky

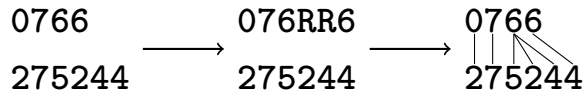
Obrázek 5: Příklady cest maticí vzdáleností, které splňují a nesplňují podmínky definice (3.2)

Postup algoritmu pro vyhledání nejoptimálnější cesty:

1. Algoritmus začíná na souřadnicích (L_A, L_B) a pokračuje směrem k první buňce.
2. Na aktuální pozici zkontroluje své sousední buňky a posune se tu s nejmenším ohodnocením.
3. V každé pozici se opakuje krok 2.
4. Algoritmus končí, když aktuální pozice je $(0, 0)$.

Pro nalezení nejkratší cesty se začíná na poslední buňce v matici vzdáleností. Na dané pozici v matici se porovnají sousední buňky směrem k začátku matice, tedy hodnoty na pozicích $(i-1, j)$, $(i, j-1)$, $(i-, j-1)$, kde i a j jsou souřadnice aktuální pozice v matici. Pozice s nejmenší hodnotou se stane aktivní. Stejný proces se opakuje, dokud se nedostaneme na pozici se všemi souřadnicemi rovných nule. Vezměme příklad z 3. Začínáme v pravém dolním rohu, ze kterého můžeme jít na souřadnice $(i, j-1)$ nebo $(i-1, j-1)$, které nabývají hodnot 9 a jsou tedy nejmenší. Z těchto dvou možností je vybrána možnost druhá, jelikož se pohybujeme ve dvou dimenzích, a tedy jeden pohyb nahoru je ušetřen, což nám dává potenciální šanci nalézt kratší cestu. Ze souřadnic $(3, 4)$ se pohybujeme dvakrát doleva, protože 7 a 3 jsou nejmenší sousedi. To samé platí pro poslední tři kroky v diagonálním směru, po kterých končíme na pozici $(0, 0)$. Všechny cesty vytvořené DTW metodou spojují první a poslední pole matice. Protože diagonála je nejkratší možná cesta, která může být nalezena, tak se snažíme zůstat co nejblíže u ní.

Výsledné zarovnání je generované ze získané cesty. Mějme cestu ukázkového příkladu na obrázku 3. Jsme na pozici $i = 0$ a $j = 0$ ohodnoceném 0, z které se pohybujeme diagonálně na pozici $(1, 1)$, jelikož jsme se dostali na pozici $(1, 1)$ diagonálním pohybem, tak jsou hodnoty na odpovídajících pozicích v sekvencích mapovány na sebe $\frac{0}{2}$. Z pozice $(1, 1)$ se pohybujeme opět diagonálně na buňku $(2, 2)$ s ohodnocením 2 a body 7, 7 jsou tedy namapovány opět na sebe $\frac{07}{27}$. To samé se opakuje pro pozici $(3, 3)$, čímž dostaneme $\frac{076}{275}$. Následně se dvakrát pohybujeme doprava, což znamená vložení dvou bodů zborcení R za aktuální bod v sekvenci A na které jsou namapovány body 2 a 4 v sekvenci B $\frac{076RR}{275\,2\,4}$. Pokud by se pozice měnila vertikálně, tak by se R vkládal do sekvence B. Jako poslední krok je posun z pozice $(3, 5)$ na $(4, 6)$ a jako v prvních třech krocích, poslední body v sekvencích jsou namapovány na sebe. Výsledné zarovnání tedy bude $\frac{076RR6}{275\,2\,4\,4}$, kde bod zborcení R označuje opakování předchozího bodu. Ve výsledku to znamená, že body 5, 2, 4 v sekvenci B jsou všechny namapovány na první výskyt bodu s ohodnocením 6 v sekvenci A.



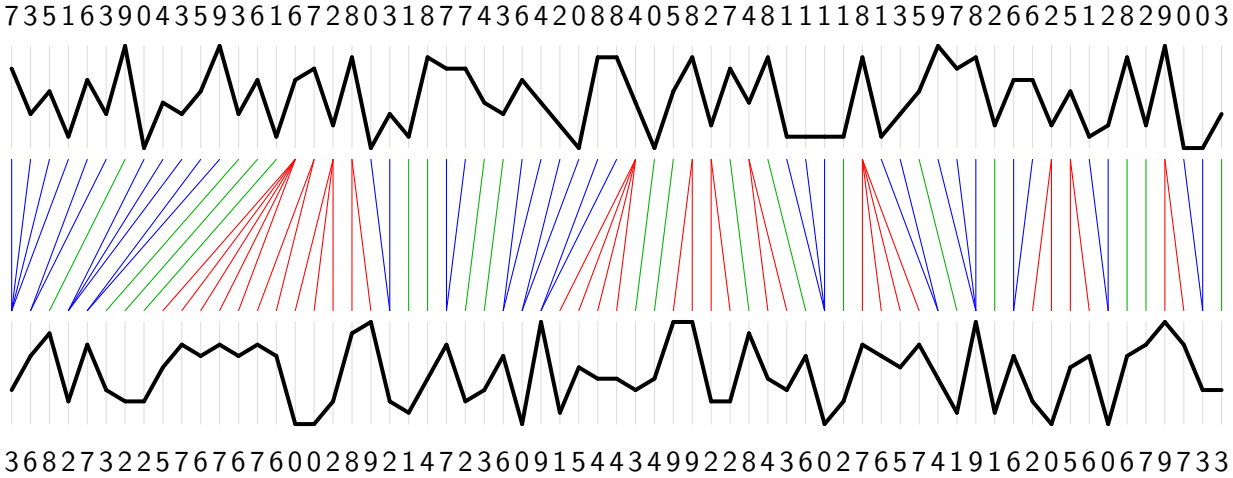
Obrázek 6: Znázornění mapování bodů mezi zarovnávanými sekvencemi

Výsledné ohodnocení zarovnání (raw score) je v poslední buňce matice vzdáleností, tedy na souřadnicích $[L_A - 1, L_B - 1]$. Čím je tato hodnota menší, tím je zarovnání efektivnější. Bohužel toto ohodnocení není použitelné k porovnání ostatních zarovnání, jelikož nemá žádnou vypovídající informaci o kvalitě daného zarovnání. Například pokud bychom zarovnali dvě delší sekvence, než v příkladě na obrázku 3, tak by skóre bylo ve většině případů automaticky větší. Díky tomu potřebuje k porovnání jednotlivých zarovnání normalizovanou hodnotu například mezi 0 a 1, která by podávala informaci o kvalitě. K získání této hodnoty můžeme například

použít tento vztah.

$$ratio = \frac{L_A + L_B}{L_{Anew} + L_{Bnew}} \tag{3.3}$$

Kde L_A a L_B jsou délky zarovnávaných sekvencí a L_{Anew}, L_{Bnew} jsou délky sekvencí po zarovnání. U DTW metody tím jsou myšleny délky sekvencí obohacených o vložené body zborcení R, jelikož DTW metoda délky sekvencí ve výsledku nemění. I když se o tomto poměru bavíme jako o podobnosti mezi zarovnávanými sekvencemi, tak spíše říká, jak moc byly dané sekvence při zarovnání upraveny. V našem případě tedy mluvíme o poměru mezi sekvencemi s body zborcení vůči sekvencím v původním tvaru.



Obrázek 7: Ukázka výstupu metody DTW nad jednorozměrnými sekvencemi

Na obrázku 7 je vidět výstup metody DTW pro jedno-dimenzionální sekvence. Modrou a červenou barvou jsou zde označeny úseky sekvencí, které se bortí do jednoho bodu sekvence druhé a naopak.

3.3 Metoda DTW nad kategoriálními sekvencemi

Metoda DTW využívá pro porovnání bodů sekvencí Euklidovu vzdálenost. Z toho důvodu ji nejde přímo použít na analýzu sekvencí kategoriálních dat, respektive pokud máme jen body symbolů bez jakékoliv jejich evaluace, tak jsou dané body neporovnatelné. Jedním ze způsobů, jak umožnit jejich analýzu, je použití tzv. substituční matice, která udává vzdálenost mezi všemi kombinacemi symbolů, které kategoriální data mohou nabýt. Například u genetických sekvencí složených z proteinů se využívá substituční matice BLOSUM, která obsahuje všechny páry proteinů a udává cenu jich editace (substituce).

Tabulka 1: Substituční matice BLOSUM62 před a po úpravě pro metodu DTW

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	4|7	12	13	13	0	12	12	11	13	12	12	12	12	13	12	10	11	14	13	11	13	12	11	15
R	-1	5|6	11	13	14	10	11	13	11	14	13	9	12	14	13	12	12	14	13	14	12	11	12	15
N	-2	0	6|5	10	14	11	11	11	10	14	14	11	13	14	13	10	11	15	13	14	8	11	12	15
D	-2	-2	1	6|5	14	11	9	12	12	14	15	12	14	14	12	11	12	15	14	14	7	10	12	15
C	0	-3	-3	-3	9|2	14	15	14	14	12	12	14	12	13	14	12	12	13	13	12	14	14	13	15
Q	-1	1	0	0	-3	5|6	9	13	11	14	13	10	11	14	12	11	12	13	12	13	11	8	12	15
E	-1	0	0	2	-4	2	5|6	13	11	14	14	10	13	14	12	11	12	14	13	13	10	7	12	15
G	0	-2	0	-1	-3	-2	-2	6|5	13	15	15	13	14	14	13	11	13	13	14	14	12	13	12	15
H	-2	0	1	-1	-3	0	0	-2	8|3	14	14	12	13	12	13	12	13	13	9	14	11	11	12	15
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4|7	13	14	10	11	14	13	12	14	12	8	14	14	12	15
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	-2	4|7	13	9	11	14	13	12	13	12	10	15	14	12	15
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5|6	12	14	12	11	12	14	13	13	11	10	12	15
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5|6	11	13	12	12	12	12	10	14	12	12	15
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6|7	15	13	13	10	8	12	14	14	12	15
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7|4	12	12	15	14	13	13	12	13	15
S	-	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4|7	10	14	13	13	11	11	11	15
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5|6	13	13	11	12	12	11	15
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11|0	9	14	15	14	13	15
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7|4	12	14	13	12	15
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4|7	14	13	12	15
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4|7	10	12	15
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4|7	12	15
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1|12	15
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1|10

Co se týká metody DTW, která vyhledává v matici vzdáleností minima, je dobré substituční matici upravit tak, aby neobsahovala záporná čísla a zaručit, že nejpodobnější body jsou ohodnoceny nejmenším číslem v celé matici. Toho se dá docílit jednoduše odečtením maximální hodnoty v substituční matici M od celé matice M v absolutní hodnotě.

$$x'_{ij} = |x_{ij} - max(M)| \tag{3.4}$$

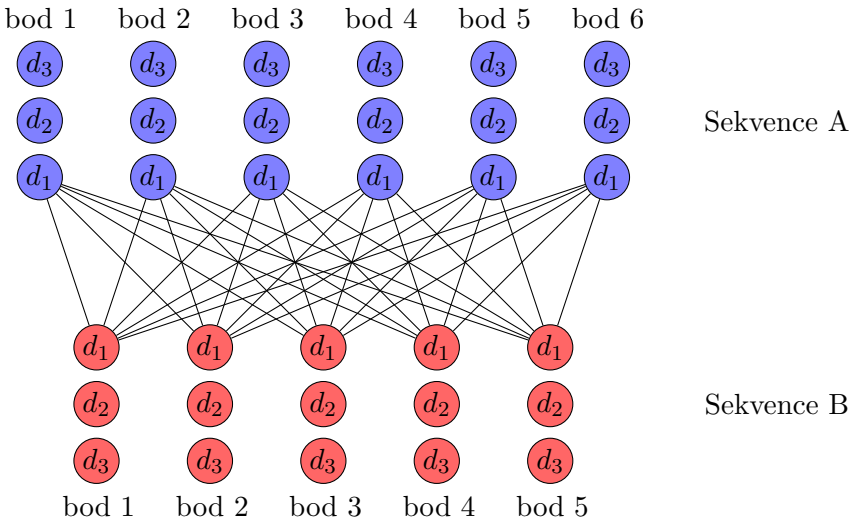
Kde x_{ij} je hodnota v substituční matici, $max(M)$ je maximální hodnota z matice M. Pro matici BLOSUM62 je nejvýhodnější substituce $(W, W) = 11$. x'_{ij} je převedená vzdálenost mezi body pro metodu DTW, i a j jsou indexy hodnot substituční matici.

V tabulce 1 je zobrazená celá substituční matice BLOSUM62[1] před a po úpravě na kladná čísla pro metodu DTW. V levé dolní diagonální matici jsou originální hodnoty, v pravé horní je pak již upravená matice pro metodu DTW.

Poznámka 1 Na diagonále v tabulce 1 je na levé straně před znakem | původní hodnota a na pravé straně nová hodnota substituční matice spočítaná podle vzorce (3.4).

[1]Substituční matice BLOSUM se používá pro měření editační ceny při zarovnávání biologických sekvencích složených z proteinů. BLOSUM [10] matice konkrétně obsahují vzdálenosti mezi aminokyselinami. Matice BLOSUM se rozlišují podle toho, z jakých dat byla daná matice vytvořena. V případě BLOSUM62 byly použity sekvence, které nabývají míru podobnosti maximálně 62%.

3.4 Multidimensional DTW



Obrázek 8: Znázornění výpočtu vzdáleností jednotlivých bodů sekvencí v MDTW metodě

Metoda MDTW umí na rozdíl od metody DTW porovnávat n-dimenzionální sekvence, tedy je schopna provádět zarovnání nad různými křivkami, zvukovými stopami atp.. Rozdíl v algoritmu mezi těmito metodami je relativně malý. Metoda DTW získává vzdálenosti v matici vzdáleností jen pomocí jednoho páru bodů. Na rozdíl o toho u metody MDTW je potřeba získat vzdálenosti mezi jednotlivými body ve všech dimenzích [1]. K tomu je využíván vzorec Euklidovské vzdálenosti.

Poznámka 2 V odborné literatuře je většinou DTW metoda zmiňována již se schopností pracovat s n-dimenzionálními sekvencemi, pokud se dále v textu budu odkazovat na metodu DTW, tak mám na mysli právě její vícerozměrnou variantu MDTW.

3.4.1 Euklidovská vzdálenost

Euklidovská vzdálenost nebo Euklidovská metrika určuje vzdálenost mezi dvěma vektory o stejném počtu prvků v n-dimenzionálním prostoru, kde jsou definovány vektory $\vec{p}$ a $\vec{q}$.

$$\vec{p} = (p_1, p_2, ..., p_k) \tag{3.5}$$

$$\vec{q} = (q_1, q_2, ..., q_k) \tag{3.6}$$

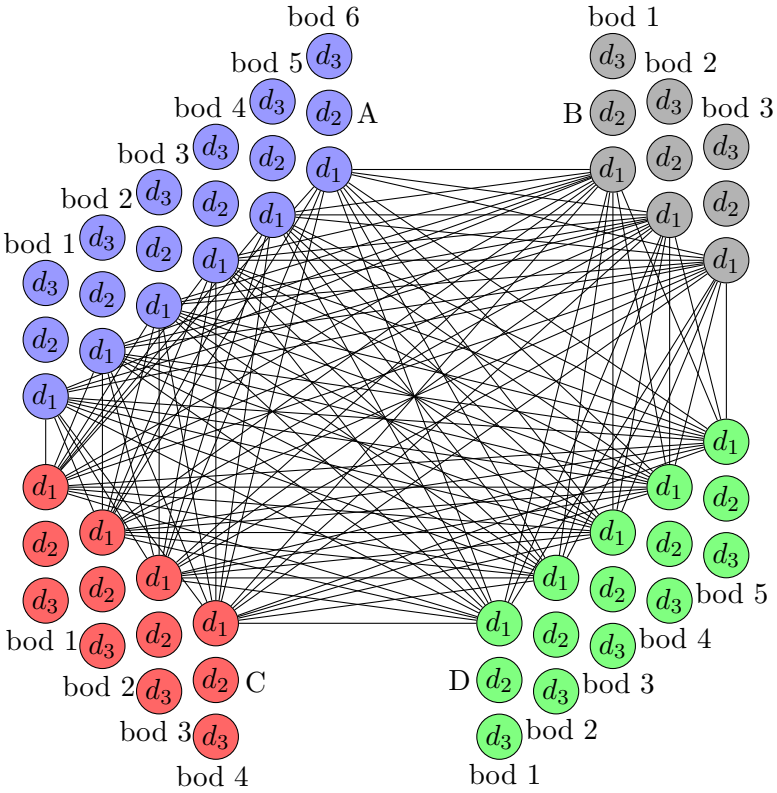
Kde d je vzdálenost mezi body p a q, k je počet dimenzí vektorů p a q. K výpočtu se používá vztah Euklidovské metriky, který je odvozený z Pythagorovy věty.

$$d(\vec{p}, \vec{q}) = \sqrt{\sum_{k=1}^{k} (p_k - q_k)^2} \tag{3.7}$$

Kde každý vektor může mít 2 až n dimenzí, ale jelikož vzdálenost mezi vektory, které mají různý počet dimenzí není měřitelná, tak všechny vektory musí být stejně rozměrné.

3.5 Multiple Multidimensional DTW

Na rozdíl od metod DTW a MDTW pochopení a implementace metody MMDTW není zdaleka tak přímočaré. Jak už bylo zmíněno, metoda MMTDW [1] [2] je časově a paměťově velmi náročná, takže i v případě kdy máme algoritmus, který teoreticky umí pracovat s neomezeným počtem sekvencí, v praxi jsme při použití MMDTW metody omezeni pracovat s jednotkami sekvencí a to navíc dané sekvence musí být extrémně krátké. Na druhou stranu jsou výsledky této metody vždy nejhůře stejně přesné jako výsledky progresivních metod. Ve většině případech má však výsledky přesnější než ostatní metody, které na úkor přesnosti dosahují lepších časových a paměťových nároků.



Obrázek 9: Znázornění výpočtu vzdáleností mezi jednotlivými body různých sekvencí v MMDTW

Definice 3.3 *N-rozměrná matice vzdáleností (distance matrix) M je* $S_1 \times S_2 \times ... \times S_n$*, kde* $S_1 = (S_{11}, S_{12}, ..., S_{1_{l_1}}), S_2 = (S_{21}, S_{22}, ..., S_{2_{l_2}}), ..., S_n = (S_{n1}, S_{n2}, ..., S_{n_{l_n}})$ *jsou sekvence.*

Postup u MMDTW metody je stejný jako u předchozích variant, tedy vytvoření a naplnění matice vzdáleností, pomocí které je následně nalezeno nejefektivnější zarovnání. Je ale potřeba upravit definici matice vzdáleností pro n-dimenzionální prostor. V metodě DTW byla velikost matice vzdáleností spočítána vynásobením délek jednotlivých sekvencí. Tady je to stejné, jen se berou délky všech sekvencí, které chceme zarovnávat.

$$size = \prod_{i=1}^{n} l_i \tag{3.8}$$

Kde n je počet sekvencí a l_i je délka dané sekvence.

Ze vzorce (3.8) je jasně vidět, že i když se pokusíme optimalizovat paměťovou náročnost zmenšením datového typu, který ukládáme do matice vzdáleností, třeba z datového typu int (4 byty) na short (2 byty), tak nám stačí přidat jednu sekvenci o délce 2 a jsme zase na původní úrovni paměti. Hodnoty samotných buněk se spočítají podobně jako v metodě DTW (MDTW), tedy využijeme vzorce (3.1) a (3.2), které upravíme pro n-dimenzionální prostředí.

$$D(i_1, i_2, ..., i_N) = d(p_1, p_2, ..., p_N) + min \begin{cases} D(i_1 - 1, i_2, ..., i_N) \\ D(i_1, i_2 1 - 1, ..., i_N) \\ \vdots \\ D(i_1 - 1, i_2 - 1, ..., i_N) \\ \vdots \\ D(i_1 - 1, i_2 - 1, ..., i_N - 1) \end{cases} \tag{3.9}$$

Kde N je počet sekvencí, $D(i_1, i_2, ..., i_N)$ je pozice v poli na souřadnicích i_1 až i_N, $d(p_1, p_2, ..., p_N)$ je vzdálenost mezi body p_1 až p_N (rozumějme, že vzdálenost je získaná porovnáním každého bodu ze sekvence se všemi ostatními, tedy $d(p_1, p_2, ..., p_N)$ obsahuje z každé sekvence právě jeden bod), min je minimální hodnota ze sousedních buněk ve směru cesty.

$$d(p_1, p_2, ..., p_N) = \sum_{a=1}^{N} \sum_{b=a+1}^{N} \sum_{k=1}^{K} || \, (p_a)_k - (p_b)_k \, || \tag{3.10}$$

Kde a je index prvního bodu, b je index druhého bodu a K udává z kolika dimenzí se skládají jednotlivé body v sekvenci (aby body v sekvencích byly porovnatelné, tak musí mít stejný počet dimenzí).

Samotná konstrukce n-dimenzionální vzdálenostní matice závisí na zvoleném programovacím jazyce, v kterém budeme metodu implementovat. Například v jazyce php či matlabu je možné dynamicky vygenerovat kód pro naplnění vzdálenostní matice a následně ho provést pomocí

funkce eval (metoda umožňující provádět kód uložený v datovém typu *string*) vykonat. Naneštěstí ve většině vyšších programovacích jazycích tohle není možné, kvůli nutnosti kompilovat kód, navíc je tento způsob výkonnostně pomalejší, jelikož daný kód je nutné interpretovat. U objektově orientovaných jazyků jako je Java, C++, C# je možnost vytvoření n-rozměrných polí pomocí objektů. Co se týká programovacího jazyku C#, v kterém je aplikace této práce implementována, tak je v něm možné použít funkci $CreateInstance(datatype, int[])$, která umožňuje vytvářet pole až o 32 dimenzích. Následně je možné přistupovat na jednotlivé buňky pomocí metod $SetValue(value, int[])$ a $GetValue(int[])$. Ukázku práce s n-rozměrnými poli v jazyce C# je možné vidět na výpise 1.

```
// proměnná musí být typu Array, jelikož nevíme jaký rank bude mít
// naše pole
Array array = Array.CreateInstance(typeof(int), sequences.Select(n => n.Count).
    ToArray());

//nastevení hodnot v poli
array.SetValue( (int)(value), coords);

//přístup k hodnotám v poli
int value = array.GetValue(coords);

//získání počtu dimenzí v poli a jejich velikosti
int[] coords = new int[array.Rank];
```

Výpis 1: Ukázka práce s n-rozměrným polem v jazyce C#

I když jsme si ušetřili práci díky funkci $CreateInstance$, tak je tu stále otázka, jak budeme generovat souřadnice pro přístup na jednotlivé pozice v poli. Díky tomu, že neznáme dopředu počet dimenzí, tak nemůžeme použít vícenásobně vnořené cykly for, tedy musíme generovat souřadnice jiným způsobem. Jednou z možností je použití algoritmu pro generování všech permutací z dané abecedy, kde naše abeceda, respektive abecedy jsou všechny hodnoty, které můžou jednotlivé souřadnice (dimenze) nabít. Jako příklad mějme čtyři sekvence, které chceme zarovnat. Každá z nich je jinak dlouhá, kde jejich délky jsou 7, 15, 20 a 13. Máme tedy 4-dimenzionální pole, kde první souřadnice může nabýt hodnot z abecedy 0-6, druhá 0-14, třetí 0-19, a poslední 0-12. Algoritmus, který generuje všechny permutace je možné vidět na výpise 2. Je zajímavé, že stejný algoritmus je možné využít pro generování variací hesel při útoku hrubou silou.

```
for (int i = 0; i < permutatios; i++){
   dimsForPoint = new List<List<int>>();
   for (int j = 0; j < sqs.Count; j++) {
      dimsForPoint.Add(sqs[j][coords[j]]);
   }
   ushort one = (ushort)Support.MEuklid(dimsForPoint);
   ushort two = GetMinNeigh(arrayND, coords);
   arrayND.SetValue( (ushort)(one + two), coords);

   int position = 0;
   do {
      coords[position] += 1;
      coords[position] %= sqs[position].Count;
   } while (coords[position++] == 0 && position < sqs.Count);
}
```

Výpis 2: Generování všech možných permutací souřadnic

3.6 Nalezení nejlevnější cesty v metodě MMDTW

Rekurzivní algoritmus pro nalezení nejlepšího zarovnání funguje podobně jako v metodě DTW, jen počet sousedních buněk v kterých se hledá minimum je vyšší. Tedy definici cesty 3.2 pro metodu DTW je potřeba upravit pro n-dimenzionální prostor, stejně jako jsme upravili definici pro matici vzdáleností.

Definice 3.4 *Cesta (warping path) pro N-rozměrnou matici vzdáleností M je sekvence $P = (p_1, p_2, ..., p_L)$, kde $p_l = (S_1, S_2, ..., S_n) \in S_1 \times S_2 \times ... \times S_n$ a $l \in \langle 1, L \rangle$ a zároveň splňující tyto podmínky.*

1. Podmínka omezenosti: $p_1 = (S_{1_1}, S_{2_1}, ..., S_{n_1})$, kde $S_{1_1}, S_{2_1}, ..., S_{n_1} = 1$ a $p_L = (L_{S_1}, L_{S_2}, ..., L_{S_n})$
2. Podmínka monotonie: $S_{1_1} \leq S_{1_2} \leq ... \leq S_{L_{S_1}}$, $S_{2_1} \leq S_{2_2} \leq ... \leq S_{L_{S_2}}, ...,$ $S_{n_1} \leq S_{n_2} \leq ... \leq S_{L_{S_n}}$
3. Podmínka velikosti kroku: $p_{l+1} = p_l \in \{(1, 0, ..., 0), (0, 1, .., 0), ..., (1, 1, ..., 0), ..., (1, 1, ..., 1)\}$ pro $l \in \langle 1, L - 1 \rangle$

Kde L_A a L_B jsou délky zarovnávaných sekvencí, A_i a B_j jsou body v sekvencích A, B.

Jelikož máme teoreticky neomezený počet sekvencí, tak nás to přivádí k otázce kolik má buňka v n-dimenzionální matici sousedních prvků? V 1D, 2D a 3D prostoru je to jednoduché

spočítat je to 2, 8 a 26. Z toho se dá odvodit jednoduchý vztah, pomocí kterého zjistíme počet sousedů i pro vyšší dimenze.

$$x = 3^n - 1 \tag{3.11}$$

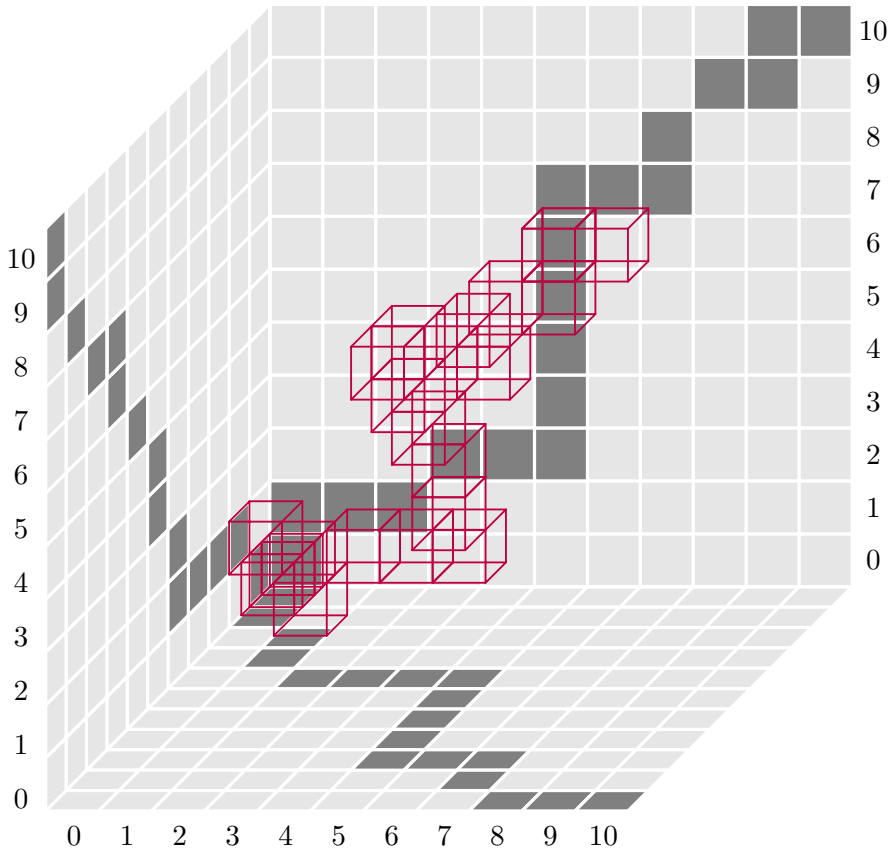
$$x = 2^n - 1 \tag{3.12}$$

Kde n je počet dimenzí (sekvencí) a x je počet sousedních buněk.

Tabulka 2: Nárůst počtu sousedních buněk se zvyšujícím počtem sekvencí (dimenzí)

n	$3^n - 1$	$2^n - 1$	n	$3^n - 1$	$2^n - 1$	n	$3^n - 1$	$2^n - 1$
1	2	1	6	728	63	11	177146	2047
2	8	3	7	2186	127	12	531440	4095
3	26	7	8	6560	255	13	>1M	8191
4	80	15	9	19682	511	14	>4M	16383
5	242	31	10	59048	1023	15	>14M	32767

Souřadnice středové buňky můžou nabývat jen tří hodnot a to -1, 0 a +1. Tedy vztah pro zjištění počtu je vztah (3.11). Jedničku odečítáme proto, abychom nepočítali samotnou středovou buňku. Hodnoty pro 1 až 15 rozměrné prostory jsou vidět v tabulce 2. Díky tomu že naše rekurzivní backtracking metoda kontroluje sousední prvky jen v jednom směru a to od poslední buňky k první, tak se vztah lehce změní. Odstraněním souřadnice -1 dostaneme vzorec (3.12), který udává počet sousedních při pohybu v jednom směru. Jak lze vidět, počet sousedních buněk se díky tomu naštěstí výrazně zmenší.



Obrázek 10: Ukázka cesty maticí vzdáleností pro tři sekvence v metodě MMDTW

Na obrázku 10 lze vidět simulaci matice vzdáleností pro 3 sekvence. Je zde dobře vidět rychlost nárůstu paměťové náročnosti. Přidáním jediné sekvence o délce 10 jsme dostali matici vzdáleností s desetkrát větší paměťovou náročností.

3.7 Progressive Dynamic Time Warping

Metoda MMDTW má exponenciální časovou a paměťovou složitost $O(n^m)$, kde n je délka sekvencí a m jejich počet. Z tohoto důvodu ji není možné použít na reálné příklady, proto zde bude vysvětlena progresivní varianta metody DTW (dále jen PDTW), která má paměťovou složitost maximálně $M(nm)$ (kde n a m jsou délky sekvencí), při libovolném počtu zarovnávaných sekvencí [12] [15].

Metoda PDTW na rozdíl od MMDTW nezarovnává všechny sekvence najednou, ale postupně. Na obrázku 11 můžeme vidět nástin toho, jakým způsobem by vypadalo zarovnání několika sekvencí. Postup vytváření zarovnání se skládá z těchto kroků:

1. Vytvoření diagonální matice, která obsahuje vzdálenosti (normalizované skóre) mezi všemi páry sekvencí, které chceme zarovnat.

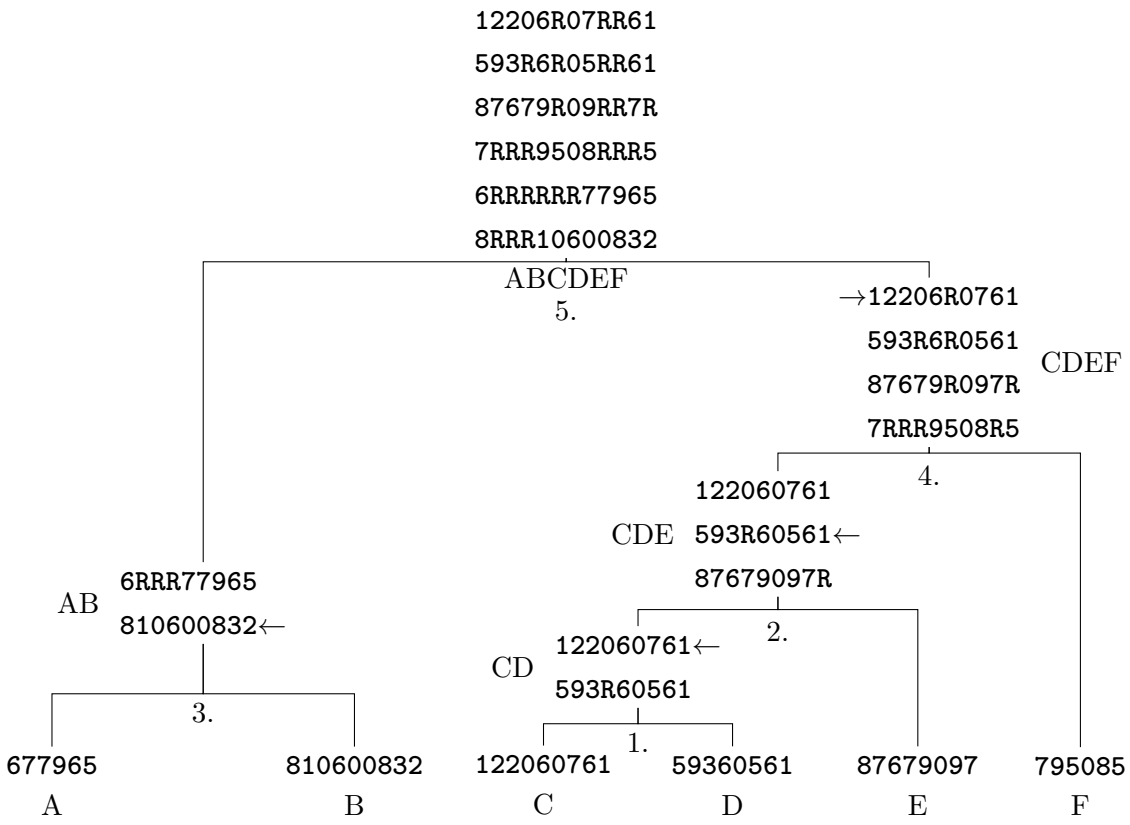
2. Pomocí diagonální matice vybereme dvě nejbližší sekvence (skupiny). Tyto dvě jsou zarovnány pomocí klasické metody DTW a sjednoceny do skupiny.

   (a) Pokud nejbližší sekvence vybrané z diagonální matice nejsou skupiny sekvencí, tak jsou zarovnány pomocí klasického DTW a sloučeny do skupiny.

   (b) Pokud se jedná o sekvenci a skupinu sekvencí nebo obě jsou skupiny, tak se napřed provede DTW nad všemi páry mezi skupinami a z nich je nejbližší pár vybrán jako reprezentativní a zarovnán DTW metodou. Sekvence z daných skupin kopírují chování svého reprezentanta.

3. Provedeme aktualizaci diagonální matice. Hodnoty pro zarovnané sekvence jsou sloučeny, aby při opětovném provádění kroku 2 byla matice aktuální.

4. Pokud existuje více než jedna skupina, tak se přejde na krok 2, jinak algoritmus končí.

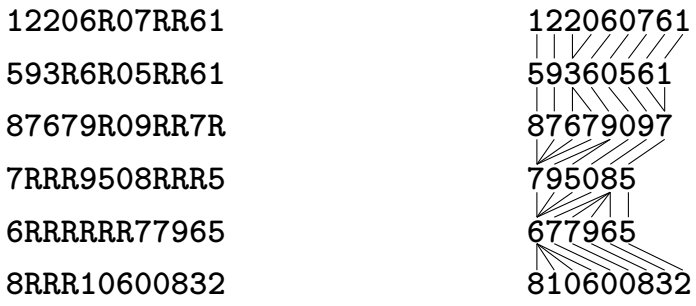


Obrázek 11: Znázornění progresivního zarovnání šesti sekvencí v metodě PDTW

Chceme zarovnat 6 sekvencí A až F, které lze vidět na obrázku 11. Na začátku vytvoří metoda PDTW diagonální matici (DM) vzdáleností všech kombinací párů sekvencí z našeho vstupu. Z výsledné DM vybere pár, který je nejpodobnější, kde v našem příkladě to jsou sekvence C a D, ty jsou zarovnány pomocí DTW metody a sloučeny do skupiny CD. Následně se přehodnotí DM matice tak, aby vzala v potaz sloučení sekvencí C a D do jedné skupiny. Z aktualizované DM matice je znovu vybrán nejpodobnější pár, což je skupina CD a E. Pokud provádíme zarovnaní nad skupinami, tak očividně tu nastává otázka, jak zarovnat skupinu se sekvencí či další skupinou. Je třeba vybrat z každé skupiny jednu sekvenci podle, které budeme provádět zarovnání. Jednoduše tedy provedeme stejnou operaci, pomocí které vytváříme DM, jen ji tentokrát neuchováváme. Vytvoříme zarovnání nad všemi páry z daných skupin pomocí klasického DTW. Zarovnání bude tedy provedeno nad sekvencemi CE a DE z nichž jsou bližší sekvence C a E, jsou tedy zarovnány a sloučeny do skupiny CDE. Při zarovnání C a E byl bod zborcení značený R vložen jen do sekvence E, díky tomu skupina CD zůstala beze změny. Pokud by při zarovnávání těchto sekvencí byl do C přidán bod zborcení R, tak by byl uměle vložen na stejnou pozici i do sekvence D, protože následuje chování vybrané vůdčí sekvence C. Tento postup je opakován, dokud nám nezbude jediná skupina. Po skupině CDE jsou sloučeny sekvence AB po níž následuje CDE s F, kde nejpodobnější s F je sekvence D. Do sekvence D byl při zarovnání s F vložen jeden bod zborcení R, na stejnou pozici je tedy vložen i do zbývají-

cích sekvencí ze stejné skupiny, jelikož následují chování lídra. Nakonec jsou zarovnané skupiny AB a CDEF, kde nejmenší vzdálenost mezi sebou mají sekvence B a C, přičemž po jejich zarovnání nám zůstane ABCDEF. Je nutné, aby zarovnání bylo prováděno vždy nad sekvencemi (skupinami), které jsou si nejbližší, tedy po každém zarovnání je DM aktualizována, tak aby reflektovala stav shlukujících se skupin sekvencí.

Při výběru dvou nejbližších sekvencí, které se stanou reprezentativním párem se využívá obou typů hodnocení, jak raw skóre, tak i normalizované skóre, obě již dříve popsané u metody DTW. Je to z toho důvodu, že normalizované skóre může v určitých situacích rovnat jedné. Například pokud jsou sekvence namapovány 1 : 1, kde délka výsledného zarovnání je stejná jako u originálních sekvencí. Jelikož sekvence po zarovnání neobsahují body zborcení jen v případě, pokud se v matici vzdáleností při hledaní nejkratší cesty pohybujeme jen diagonálním směrem. V takovém případě je použito raw skóre, které vyjadřuje přesněji vzdálenost mezi dvěma konkrétními sekvencemi bez vztahu k ostatním. Pokud je použito raw skóre, tak záleží na použité metodě, jestli hledáme nejmenší či největší raw skóre. U metody DTW, respektive PDTW hledáme minimální skóre, ale například u LCSS metody, která bude vysvětlena později, je to maximum.



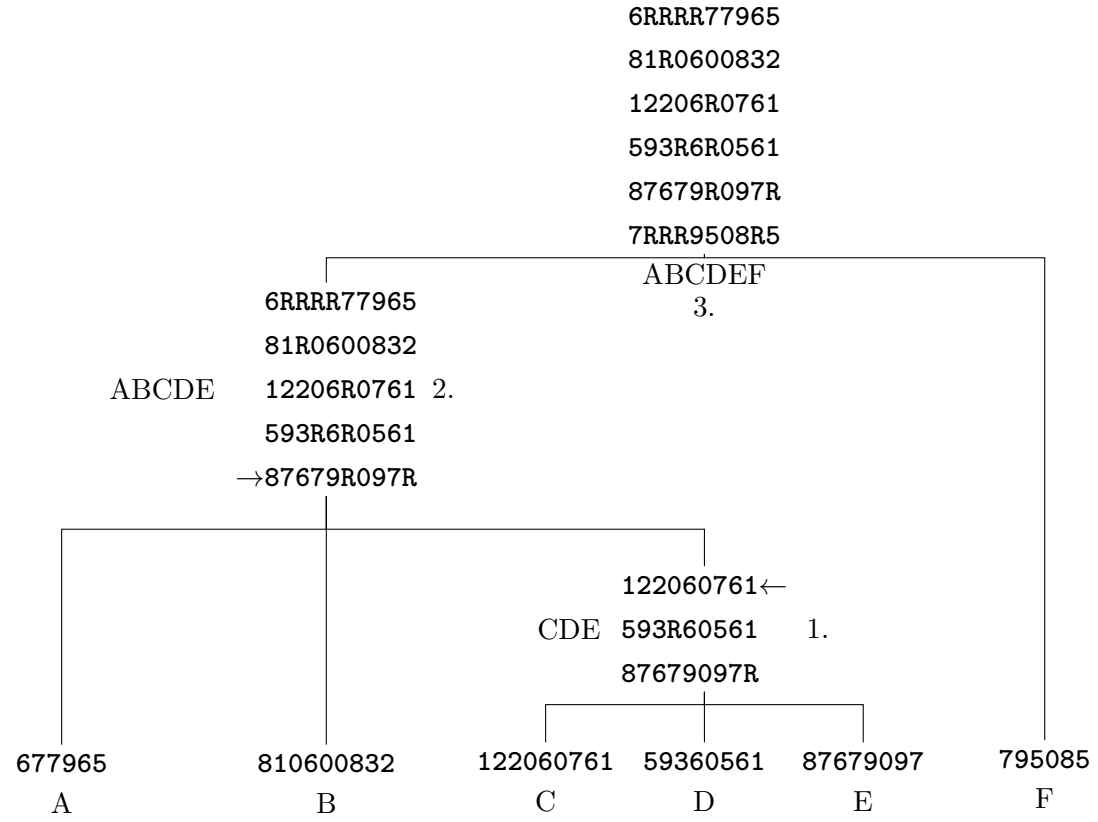
(a) A (b) B

Obrázek 12: Znázornění mapování bodů šesti sekvencí v metodě PDTW

3.8 Modular Progressive Dynamic Time Warping

Poslední varianta DTW metody, která bude v této práci vysvětlena se jmenuje Modular Progressive Dynamic Time Warping (MPDTW). Kombinuje metody MMDTW a PDTW, za účelem se přiblížit k přesnosti výsledků metody MMDTW a snížení časové a paměťové náročnosti, čehož bylo docíleno progresivní variantou. Postup MPDTW metody je prakticky totožný s postupem PDTW metody, jen se pro zarovnání sekvencí nevyužívá klasická DTW metoda pro analýzu párových sekvencí, ale MMDTW varianta. Za tímto účelem přidává tato metoda parametr γ (cluster size), který udává počet sekvencí, respektive skupin sloučených najednou. U PDTW metody to vždy byly 2 skupiny a u MMDTW proběhlo vždy jen jedno zarovnání nad všemi sekvencemi najednou. Na rozdíl od nich metoda MPDTW tento parametrizuje. Tímto je dosaženo jednoduché, ale poměrně účinné optimalizace nepřesnosti PDTW metody a časové i paměťové

složitosti MMDTW metody. Dá se říct, že MPDTW perfektně balancuje mezi těmito dvěma metodami, jelikož se pomocí parametru γ dá MPDTW dostat na časovou úroveň PDTW metody, tak na přesnost MMDTW, samozřejmě za cenu poklesu přesnosti výsledků, respektive nárůstu času a paměti.



Obrázek 13: Znázornění progresivního zarovnání šesti sekvencí metodou MPDTW

Náhled vytváření guide tree u metody MPDTW lze vidět na obrázku 13. Jak už bylo vysvětleno sekvence se seskupují podle toho, jak je nastaven parametr γ. V našem příkladě je nastaven na hodnotu 3. Na začátku algoritmu je vytvořena diagonální matice (DM), která uchovává vzdálenosti mezi všemi sekvencemi. Pokračujeme vyhledáním 3 nejbližších sekvencí, k čemuž je využita pravě vytvořena DM. Nejbližší sekvence v první iteraci jsou C, D a E, které jsou tedy zarovnány pomocí metody MMDTW do skupiny CDE. Na konci první iterace je aktualizována diagonální matice, aby brala zarovnanou skupinu CDE jako jednu hodnotu. V druhé iteraci se postup opakuje, kde nejbližší sekvence (skupiny) jsou A, B a CDE. Pokud zarovnáváme skupiny, tak podobně jako u PDTW metody je potřeba nalézt nejbližší trojici (podle parametru γ). Vytvoříme tedy zarovnání všech kombinací sekvencí z daných skupiny, které jsou ABC, ABD a ABE. Nejbližší jsou sekvence A, B a C, které jsou tedy zarovnány a sloučeny do skupiny ABCDE. Body zborcení R, které byly přidány do sekvence C jsou vloženy na stejnou pozici i do zbylých sekvencí ze skupiny CDE. Poté je opět aktualizována DM matice, aby reflektovala sloučení sek-

vencí A, B a skupiny CDE. V poslední iteraci máme jen skupinu ABCDE a sekvenci F, tedy provede se zarovnání nad všemi kombinacemi párů pomocí klasické metody DTW (MPDTW s $\gamma = 2$). Nejbližší zarovnání se použije a sekvence, která byla vybrána jako reprezentativní ze skupiny ABCDE určuje chování zbylých sekvencí patřících do této skupiny. Tímto je vytvořené konečné zarovnání ABCDEF.

Na první pohled se může zdát, že počet zarovnání je menší než u metody PDTW, což není pravda. U DTW metody se při zarovnávání skupin provádí $S_A \cdot S_B$ zarovnání, kde S_A a S_B jsou velikosti skupin A a B. U MPDTW je počet zarovnání závislý na parametru γ, tedy počet zarovnání je:

$$mmdtws = \prod_{i=1}^{\gamma} S_i \tag{3.13}$$

Kde mmdtws je počet zarovnání provedených při zarovnávání skupin v metodě MPDTW a S_i je velikost sjednocované skupiny.

3.9 Srovnání metod PDTW, MMDTW a MPDTW

Porovnání MMDTW metody s její progresivní variantou, která se k té pravé efektivitou a přesností výsledků ve většině případů jen blíží je poměrně problematické. Jak už bylo zmíněno pravá mdtw metoda má exponenciální časovou složitost, díky čemu ji lze prakticky použít pouze na malé testovací příklady, které jsou díky jejich minimálnosti zkreslené.

V naprosté většině případů má mmdtw metoda lepší výsledky, ale existují případy, kdy zarovnání pomocí pdtw metody vykazuje „lepší“ výsledky. Je to způsobeno chybou, která je při použití progresivního zarovnání nutně vytvářena. Jak bylo vysvětleno výše, pdtw metoda vybírá ze zarovnávaných skupin, které jsou vytvářeny při generování guide tree reprezentativní sekvence, podle kterých se dané zarovnání řídí. Tímto se ztrácí vliv těch sekvencí, které kopírují chování podle vedoucí. Je nutné si uvědomit, že tato chyba způsobuje oba případy, jak obvyklé horší, tak i vzácně zdánlivé lepší zarovnání, které je ve skutečnosti také horší. Přesněji řečeno, pokud má PDTW metoda lepší výsledky, tak se při progresivním zarovnání ztratil vliv těch sekvencí, které by výsledek zhoršily. Naopak u horších výsledků se ztratil vliv sekvencí, které by zarovnání zlepšily. Tento efekt graduje, čím více sekvencí zarovnáváme. Je to tím, že čím větší skupiny jsou zarovnávány, tím více informací se ztrácí. Například zarovnáme dvě skupiny, kde každá obsahuje 10 sekvencí, tak jen jeden pár ovlivňuje výsledné zarovnání a informace ze zbylých 18 je ztracena.

Přibližný počet provedených zarovnání pomocí párových DTW u PDTW metody lze spočítat jako součet $gt = \frac{n^2+n}{2} - 2$, udávající počet dtw zarovnání provedených při seskupování *guide tree* a $dm = \frac{n^2+n}{2}$, které určuje počet DTW při vytváření DM, tedy celkový počet získán jako $dtws = n^2 + n - 2$ s tím, že $n \geq 2$. Z toho lze získat počet buněk, které pdtw metoda počítá během jejího průběhu. Například mějme 7 sekvencí o délce 14, tedy celkový počet buněk pro

matice vzdáleností je získán $size = (n^2 + n - 2)m^2$. V porovnání s tím počet buněk počítaných mmdtw metodou je $size = m^n$, kde n je počet sekvencí a m je délka sekvencí.

Pro metodu MPDTW je počet provedených zarovnání metodou MMPDTW již trochu komplikovanější získat. Stejně jako u PDTW metody vytváříme DM matici, ke které přičítáme počet zarovnání při vytváření stromu. Přibližný počet zarovnání lze získat jako:

$$mmdtws = \sum_{i=1}^{\frac{n}{\gamma-1}} (\gamma^i)^{\gamma-1} n \tag{3.14}$$

Počet buněk který je spočítán během metody MPDTW, je pak získán vynásobením počtu provedených zarovnání při vytváření DM, velikostí matic m^2. To samé se provede se vzorcem (3.14), který vynásobíme m^γ, kde velikost této matice závisí na velikosti námi nastavovaného parametru γ.

$$size = (\frac{n^2 + n}{2})m^2 + (\sum_{i=1}^{\frac{n}{\gamma-1}} (\gamma^i)^{k-1})m^\gamma \tag{3.15}$$

Kde *mmdtws* je počet zarovnání provedených MMDTW metodou v průběhu MPDTW metody, *size* je počet ohodnocených buněk v matici vzdáleností, které bylo nutné spočítat, n je celkový počet sekvencí zarovnávaných MPDTW metodou, m je délka sekvencí, γ určuje po kolika sekvencích jsou sekvence shlukovány.

Tabulka 3: Srovnání Progresivní DTW a MMDTW metody

		PDTW			MMDTW		
Délka s.	Počet s.	ratio	Buněk	Čas	ratio	Buněk	Čas
5	3	0,714	250	1ms	0,83	125	1ms
5	5	0,83	700	1ms	0,83	3125	3ms
5	10	0,45	2700	1ms	0,83	9765625	13min 5s
20	3	0,74	400	2ms	0,8	8k	42ms
20	4	0,58	7200	3ms	0,74	160k	420ms
20	5	0,58	11,2k	4ms	0,714	3,2M	17s
50	3	0,666	25k	15ms	0,735	125k	176ms
50	4	0,625	45k	30ms	0,71	6,25M	17s
50	5	0,574	70k	50ms	0,64	312,5M	30min
100	4	0,628	180k	35ms	0,68	100M	5min
200	3	0,692	400k	60ms	0,74	8M	12s
200	4	0,617	720k	144ms	-	1600M	-
1000	3	0,667	1120k	1,7s	-	1000M	-

Jak lze pozorovat z tabulky 3, metoda MMDTW má sice přesnější (efektivnější) výsledky, ale jelikož je výrazně náročnější, tak jsou v praxi z nutnosti používané vícenásobné metody, s rozumnou časovou složitostí za cenu méně přesných zarovnání. S exponenciální časovou složitostí si v mnoha případech neporadí ani superpočítačová centra. V tabulkách 4 a 5 jsou výsledky testů metody MPDTW, která kombinuje přístup PDTW a MMDTW metod.

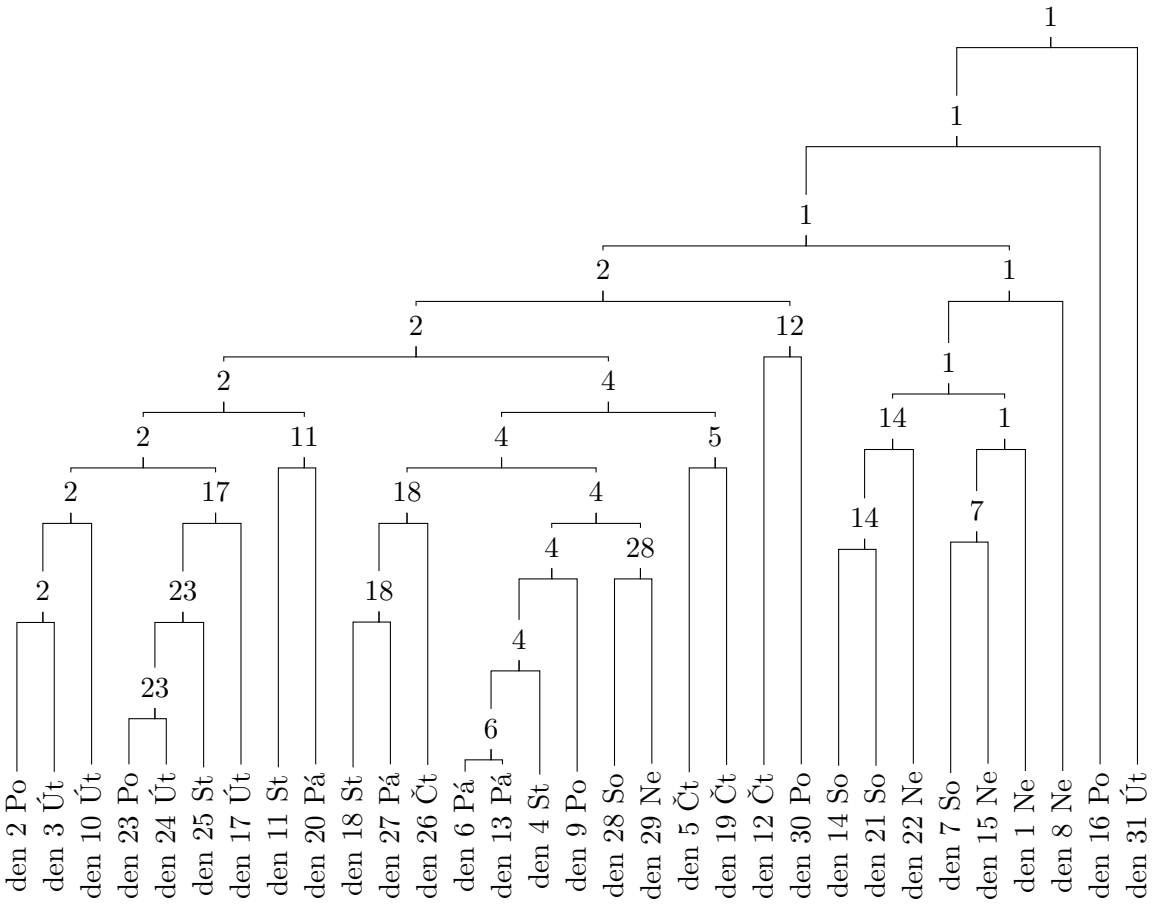
Tabulka 4: Srovnání PDTW a MPDTW metody

		$\gamma = 2$ (PDTW)		$\gamma = 3$		$\gamma = 4$	
Délka s.	Počet s.	Ratio	Čas	Ratio	Čas	Ratio	Čas
3	5	0,75	1ms	1	1ms	0,75	1ms
3	10	0,75	2ms	0,75	2ms	1	5ms
5	5	0,83	1ms	0,83	1ms	0,71	4ms
5	10	0,5	4ms	0,714	7ms	0,71	30ms
10	5	0,58	3ms	0,71	10ms	0,58	50ms
10	10	0,52	10ms	0,55	55ms	0,769	740ms
10	20	0,4	80ms	0,39	275ms	0,55	10s
10	40	0,416	200ms	0,52	8s	0,52	3m 37s
50	5	0,62	50ms	0,65	1,2s	0,617	36s
50	10	0,43	240ms	0,476	7s	0,51	6m 27s
50	20	0,34	1s	0,387	50s	-	9h+
50	40	0,295	7s	0,33	28m	-	-
100	5	0,578	220ms	0,63	11s	0,628	10m
100	10	0,469	1s	0,469	23s	0,562	3h 13m
100	20	0,34	6s	0,42	12m 23s	-	-
100	40	0,26	33s	0,299	2h 57m	-	-
500	5	0,55	5s	0,608	24m	-	-

Výsledky testů metody MPDTW je možné vidět v tabulkách 4 a 5. Lze vidět, že výsledky mají poměrně vysokou úroveň fluktuace zejména pro krátké sekvence. U delších sekvencí již lze pozorovat rostoucí tendencí ratia.

Tabulka 5: Srovnání PDTW a MPDTW metody

		$\gamma = 5$		$\gamma = 6$	
Délka s.	Počet s.	Ratio	Čas	Ratio	Čas
3	5	0,75	2ms	-	-
3	10	0,75	5ms	0,75	15ms
5	5	0,83	20ms	-	-
5	10	0,71	100ms	0,83	300ms
10	5	0,71	900ms	-	-
10	10	0,55	4,5s	0,588	23s
10	20	0,5	22ms	0,66	3m
10	40	0,588	50m	-	-
50	5	0,714	1h 10m	-	-



Obrázek 14: Znázornění shlukování sekvencí pomocí metody PDTW nad reálnými daty

Na obrázku 14 lze vidět shlukování pomocí metody PDTW nad sekvencemi naměřenými v reálném provozu. Jedná se o měření rychlosti provozu během jednoho měsíce na vybraném místě dálniční sítě ve Velké Británii[2]. Hodnoty byly měřeny na deseti vybraných místech zaznamenávány během celého dne v intervalu 15 minut. Je zde dobře vidět, že metoda PDTW shlukuje nejpodobnější sekvence jako první. Dny jako pondělí a úterý jsou často spojeny první. Stejné chování lze pozorovat u párů jako jsou pátek a středa, sobota a neděle. Nakonec stejné dny v týdnu jsou také často sjednoceny jako první.

Je dobré si uvědomit, že strom vytvořený progresivním algoritmem se může lišit, pokud jsou vstupní sekvence předány algoritmu v jiném pořadí. Vnitřní algoritmus vyhledává maximální ratio jako $if\ maxRatio < ratio\ then\ maxRatio = ratio$, tedy pokud je ratio pro více páru stejné, tak se použije první nalezené maximum. Pokud by podmínka implementace byla $maxRatio <= ratio$, tak by se priorita převrátila. Tato vlastnost se nemusí projevit jen různými stromy, ale i jiným výsledným zarovnáním a tedy i rozdílnými podobnostmi. Čím více sekvencí shlukujeme a čím kratší jsou, tím je větší pravděpodobnost různých výsledných zarovnání při jiném pořadí vstupních sekvencí.

Tabulka 6: Metoda PDTW a MPDTW nad daty rychlosti provozu na vybraných místech v Anglii

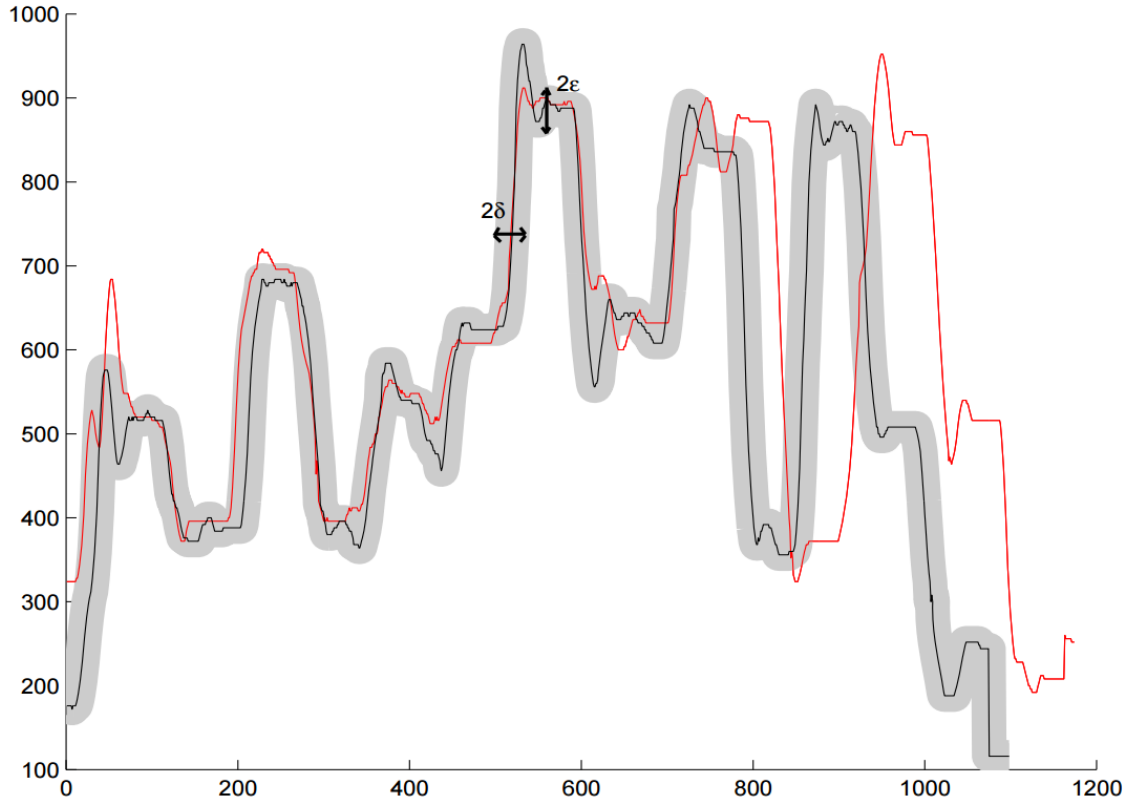
Místo	Počet s.	PDTW	MPDTW $\delta = 3$
1	31	0,2704	0,2526
2	31	0,2430	0,3057
3	31	0,2109	0,2711
4	31	0,2068	0,2689
5	31	0,1975	0,2201
6	31	0,1951	0,2394
7	31	0,1846	0,2227
8	31	0,2004	0,2142
9	31	0,2181	0,3254
10	31	0,2033	0,3189

V tabulce 6 pak lze vidět výsledky zarovnání dat zobrazených na obrázku 14 spolu s výsledky dalších 9 analyzovaných míst.

[2]Jedná se o data měřená od roku 2009 na všech dálnicích a silnicích kategorie 'A' spravovaných Highways Agency, v Anglii známou jako Strategic Road Network [17]. Měřená místa jsou anonymní z důvodu zamezení zaujatosti při jejich analýze.

4 Longest Common Subsequence

I když metoda DTW a spousta jejích variant je využívaná v mnoha vědních oborech, tak má i své nevýhody, díky kterým ji není v hodné použít na všechny typy dat. Jednou z nich je kladení příliš velkého důrazu na lokální extrémy, jelikož nutně mapuje každý bod ze zarovnávaných sekvencí a díky tomu může být výsledné zarovnání těžce pokřivené, respektive neefektivní. Jako alternativa k metodě DTW zde bude vysvětlena metoda Longest Common Subsequence [5] [6] [7] [8], dále jen LCSS, která zmíněný problém buď řeší nebo alespoň minimalizuje.



Obrázek 15: Znázornění parametrů δ a ε v metodě LCSS [8]

4.1 Longest Common Subsequence

Metoda LCSS přidává do algoritmu dva parametry δ a ε, které dovolují upravit hrubost s jakou jsou body sekvencí na sebe namapovány. Parametr δ vytváří tzv. posuvné okno (sliding widnow či warping window), které udává o kolik se jedna sekvence může posunout vpřed či zpět v čase vůči druhé sekvenci. Naproti tomu ε stanovuje práh, kdy se body sekvencí stále rovnají a kdy již jsou příliš rozdílné. Kombinací obou parametrů je vytvořen pás v prostoru, kde je podobnost elementů považována za přípustnou, a tedy dané body mohou být mapovány. Pokud je jedna

z porovnávaných sekvencí, respektive bodů mimo dané pásmo, tak jsou dané lokální extrémy vynechány z výsledného zarovnání. Znázornění parametrů je vidět na obrázku 15.

Poznámka 3 Metoda Longest Common Subsequence (LCSS) je často zaměňována za metodu Longest Common Substring (LCS). Rozdíl mezi těmito metodami je ten, že metoda LCS hledá podřetězec, který musí být souvislý. Na rozdíl od toho LCSS metoda tuto podmínku nevyžaduje.

Výpočet LCSS probíhá velmi podobně jako u metody DTW. Postup jednotlivých kroků je stejný.

1. Vytvoření a inicializace matice vzdáleností (distance matrix) a její následné naplnění.
2. Vyhledání cesty zarovnání (warping path), respektive určení optimálního zarovnání daných sekvencí.
3. Analýza výsledků, určení podobnosti sekvencí a jejich vizualizace.

Liší se ve způsobu, jakým jsou získány hodnoty v matici vzdáleností. K tomu jsou použity vzorce (4.1) a (4.2).

$$D(i,j) = \text{ if } d(i,j) \text{ then } D(i-1,j-1)+1 \text{ else } max \begin{cases} D(i-1,j) \\ D(i,j-1) \end{cases} \tag{4.1}$$

$$d(i,j) = \text{if } \mid i-j \mid \leq \delta \text{ and } \mid A_i - B_j \mid \leq \varepsilon \text{ then } \textit{true} \text{ else } \textit{false} \tag{4.2}$$

Jak lze vidět, od metody DTW se liší tím, že v sousedních buňkách je hledáno maximum a hodnoty předcházejících hodnot v matici se zvyšují jen v případě, kdy právě porovnávané body v sekvencích jsou ekvivalentní, respektive splňují podmínku ε. Mějme Sekvence A a B, kde L_a a L_b jsou jejich délky. Na začátku LCSS metody je vytvořena matice vzdáleností o velikosti $L_a \cdot L_b$, která je inicializovaná na 0 na všech pozicích. Sekvence A je 0766 a B 275244. Matici vzdáleností pro ně lze vidět na obrázku 16.

i	j	0	1	2	3	4	5	6
		B	2	7	5	2	4	4
0	A	0	0	0	0	0	0	0
1	0	↑ 0	← 0	0	0	0	0	0
2	7	0	↑ ← 0	1	1	1	1	1
3	6	0	0	↖ 1	2	2	2	2
4	6	0	0	1	↖ 2	← 2	← 2	← 2

(a) $\varepsilon = 1$, $\delta = \infty$, $shoda = 2$

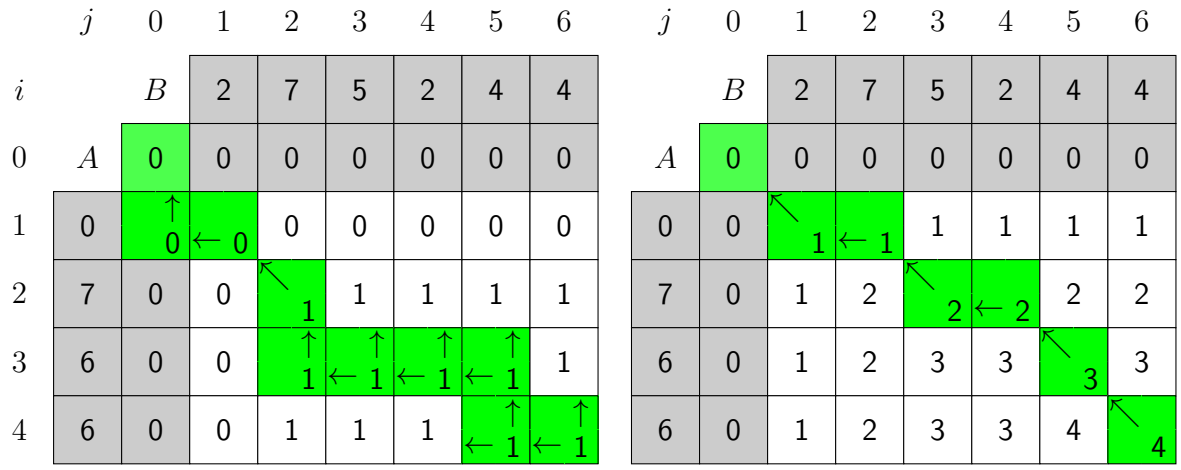
$$| A_i - B_j | \leq \varepsilon \wedge \quad | i - j | \leq \quad \delta$$
$$| 6 - 5 | \leq \varepsilon \wedge \quad | 4 - 3 | \leq \quad \infty$$
$$D(3,4) = D(4-1, 3-1) + 1 = 2$$

(b) Výpočet vzdálenosti pro D(4,3)

Obrázek 16: Ukázka výpočtu matice vzdáleností pro metodu LCSS

Matice vzdáleností je plněna od první buňky $(1,1)$ po (L_a, L_b). Plnění je prováděno po řádcích či sloupcích, kde jednotlivé výpočty jsou závislé na výpočtech z předchozích kroků. Každá hodnota, která je spočítána a uložena do matice, potřebuje ke svému výpočtu hodnoty ze sousedních prvků ze směru, z kterého je matice plněna. Konkrétně to jsou hodnoty ze souřadnic $(i-1, j), (i, j-1), (i-1, j-1)$, kde jedinou výjimkou jsou buňky s $i = 1$ nebo $j = 1$, jelikož ty technicky nemají všechny požadované sousední buňky nutné pro výpočet. V implementaci se pro ošetření této chyby většinou vytváří matice vzdáleností o 1 větší v obou směrech. Uměle přidané pozice jsou vidět na obrázcích 16 a 17 na indexech $i = 0$ a $j = 0$.

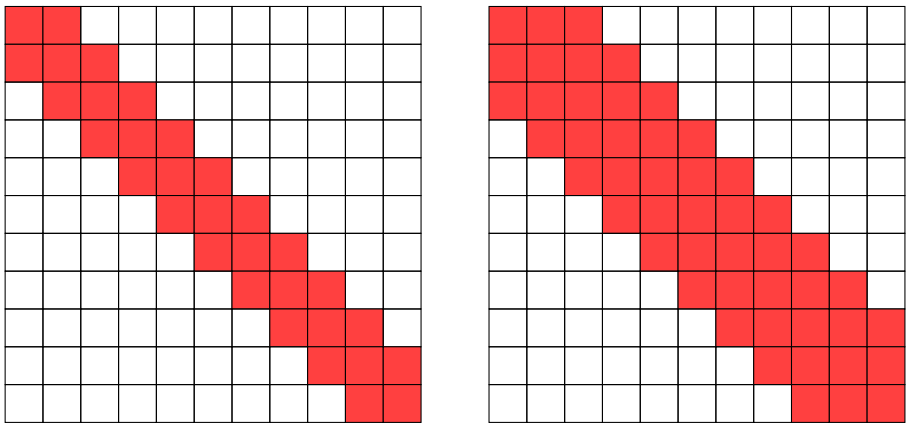
Příklad výpočtu hodnot matic lze vidět na obrázku 16. Hodnota pro pole $(4, 3)$ je získána jako vzdálenost mezi A_4 a B_3, tedy $| 6 - 5 | = 1$. Parametr ε je v našem příkladě nastaven na 1, čímž je podmínka ε splněna. Ohodnocení pro aktuální pozici je tedy získáno jako součet pozice $(3, 2)$ + 1. Pokud by podmínka pro parametr ε nebyla splněna, tak by se vybrala maximální hodnota ze zbylých již ohodnocených pozic, které sousedí s aktuální pozicí. Parametr δ byl v našem příkladu neomezený, tedy nemuseli jsme se o něho starat. Parametr δ zajišťuje, aby výsledná cesta nebyla příliš odchýlena od diagonály, tedy že čím je δ menší, tím je cesta přímější. Elementy v sekvencích jsou eliminovány ze zarovnání jen při horizontálních či vertikálních pohybech, tedy čím je δ menší a ε větší, tím je jich odstraněno méně a naopak. Tento efekt je klíčovou vlastností LCSS metody, jelikož odstraněné body jsou většinou buď lokální extrémy nebo jejich časová dilatace je příliš velká. V obou případech by nepříjemně pokřivovaly dané zarovnání. U dat, které mají vysokou míru podobnosti se většinou nastavuje ε malé a naopak je tomu u dat s velkým rozptylem hodnot.



(a) $\varepsilon = 0$, $\delta = \infty$, $shoda = 1$ (b) $\varepsilon = 2$, $\delta = \infty$, $shoda = 4$

Obrázek 17: Ukázka výpočtu matice vzdáleností metody LCSS pro různé hodnoty ε

Na obrázku 17 je možné vidět matice vzdáleností pro stejné dvě sekvence, ale s nastavením parametru ε na hodnoty 0 a 2. Jak lze vidět, cesta pro případ a je delší oproti variantě b, která je kratší. Další zvětšování parametru ε již nemá smysl, jelikož v případě b jsme již dosáhli nejkratší možné cesty. Samozřejmě pro jedno konkrétní nastavení parametrů může existovat více nejkratších cest, která je z nich vybrána závisí na konkrétních implementacích daného algoritmu. Například jedna implementace může upřednostňovat pohyb vlevo v matici vzdáleností, kde jiná upřednostňuje pohyb nahoru.



(a) $\delta = 1$ (b) $\delta = 2$

Obrázek 18: Efekt parametru δ na výslednou cestu

Jak už bylo řečeno, parametr δ vytváří tzv. warping window pro určení přijatelných časových posunů. Grafické znázornění vlivu na matici vzdáleností je vidět na obrázku 18. Pokud provádíme zarovnání nad sekvencemi s výrazně různou délkou, tak použití δ je poměrně problematické. Představme si, že máme dvě sekvence o délce 10, 100 a parametr $\delta = 10$. V tomto případě se body

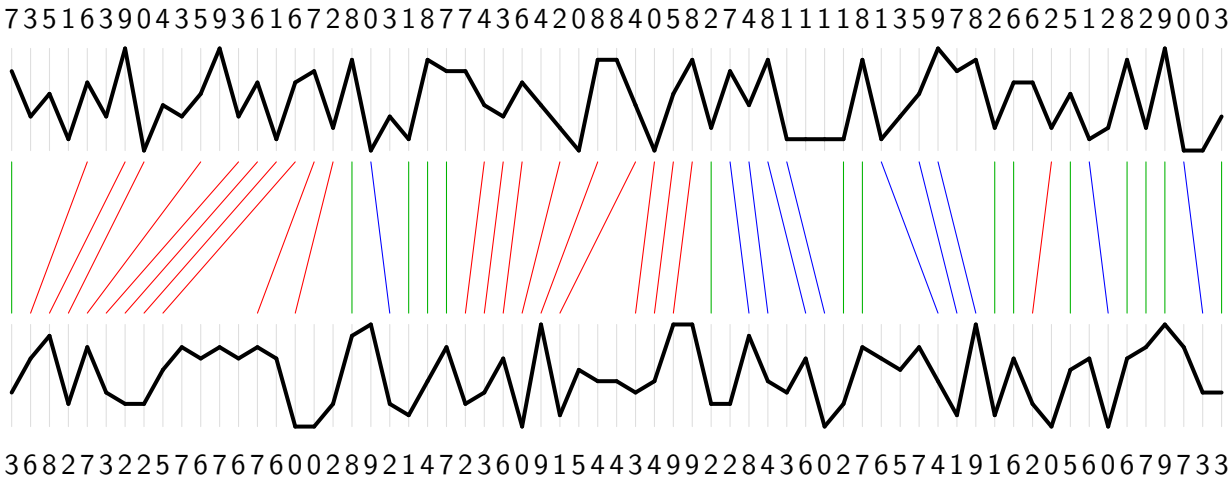
druhé sekvence mohou mapovat s první jen na první desetině své délky a zbytek sekvence bude ignorován. Globální zarovnávání je ve většině případů v hodné pro sekvence o stejné nebo alespoň podobné délce. Pokud chceme zarovnávat sekvence o různých délkách, tak lokální zarovnání je v takovém případě často vhodnější.

Cesta v matici vzdáleností je definována stejně jako v metodě DTW (3.2). Rozdíl je ale ve způsobu, jakým je daná cesta zkonstruována. Na rozdíl od DTW metody, kde je mezi sousedními prvky hledáno minimum, metoda LCSS se posouvá diagonálním směrem, pokud se hodnoty v sekvencích A a B na pozicích i a j rovnají. V ostatních případech je vyhledáno maximum ze sousedních pozic $(i-1, j)$ a $(i, j-1)$.

Vezměme příklad z obrázku 17 případ b. Začínáme na pozici $(4, 6)$ s ohodnocením 4, z které se posouváme na pozici $(3, 5)$, jelikož parametr ε je nastaven na 2, tedy $A_4 = B_6$. Stejným způsobem se posuneme na pozici $(2, 4)$. Následně hodnoty v sekvencích na aktuální pozici se nerovnají, jelikož $|A_2 - B_4| = |7 - 2| = 5 > \varepsilon = 2$, tedy posouváme se doleva na pozici $(2, 3)$. Pokud by pozice $(1, 4)$ měla ohodnocení stejné, tak bychom mohli použít kteroukoli z nich. Následuje diagonální posun na pozici $(1, 2)$ jelikož $|7 - 5| \leq 2$. Pokračujeme posunem doleva a končíme diagonálním pohybem na pozici $(0, 0)$, protože první body v sekvencích s ohodnocením 0 a 2 jsou považovány za stejné. Při každém pohybu doleva je vynechán bod ze sekvence B a to samé platí pro sekvenci A při vertikálním pohybu. Ve výsledném zarovnání budou tedy chybět body B_2 a B_4.

Poznámka 4 Parametr δ je možné využívat, také jako omezení prostoru v jakém je cesta hledána. Cesta je považována za efektivní, pokud se příliš neodchyluje od diagonály, tedy v některých situacích může prohledávání celé diagonální matice být zbytečné, protože zarovnání které je příliš drahé bychom stejně nepřijali. Záleží na konkrétních datech, jak moc velké toto omezení bude. Samozřejmě pokud je omezen prostor, v kterém je hledaná cesta, tak hodnoty pro zbylou část matice nemusíme počítat. Tímto velice jednoduše můžeme zmenšit časovou náročnost dané metody.

Pokud analyzujeme kategoriální data s metodou LCSS, tak vzdálenosti v substituční matici převádíme (3.4) na kladné stejným způsobem jako u metody DTW.



Obrázek 19: Vizualizace metody LCSS pro dvě jednorozměrné sekvence

Na obrázku 19 lze vidět výstup metody LCSS. Červená a modrá barva označuje situaci, kdy jedna sekvence předbíhá druhou a naopak, zeleně jsou pak zvýrazněny synchronizované body. Body, které jsou vynechány, nesplnily podmínku (4.2).

Úroveň podobnosti mezi párem sekvencí je u LCSS metody počítána trochu odlišným způsobem než u metody DTW (3.3). Jelikož poslední pole v matici vzdálenosti obsahuje délku nalezené subsekvence, tak ratio lze získat jako:

$$ratio = \frac{M(L_A, L_B)}{(L_A + L_B)/2} \tag{4.3}$$

Kde *ratio* je podobnost mezi sekvencemi A a B, M je matice vzdáleností, L_A a L_B jsou délky sekvencí A a B.

4.2 Multidimensional Longest Common Subsequence

Rozšíření běžné LCSS metody pro n-dimenzionální data funguje oproti metodě DTW jiným způsobem. U DTW metody jsme sčítali vzdálenosti mezi body ve všech dimenzích, čímž jsme dostali vzdálenost mezi n-dimenzionálními body v dvourozměrném prostoru. Na rozdíl od toho metoda LCSS vyžaduje, aby každá vzdálenost mezi jednotlivými dimenzemi vyhovovala podmínce ε.

$$\vec{p} = (p_1, p_2, ..., p_k) \tag{4.4}$$

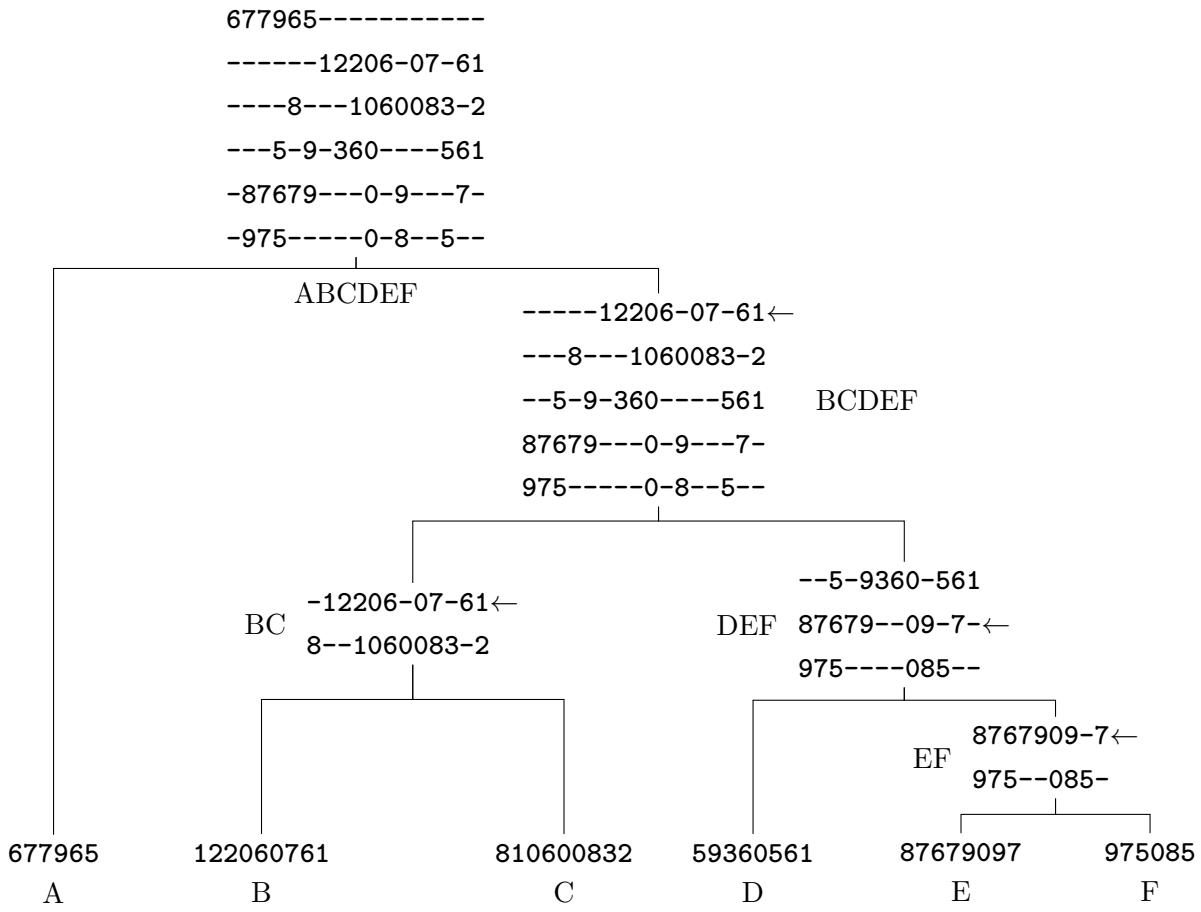
$$\vec{q} = (q_1, q_2, ..., q_k) \tag{4.5}$$

$$(|p| = |q|) \wedge (|p - q| = d = (d_1, d_2, ..., d_k), \text{kde } \forall d_i \leq \varepsilon) \tag{4.6}$$

Kde p a q jsou body v sekvencích, k je počet dimenzí bodů sekvencí, d je vzdálenost mezi jednotlivými dimenzemi.

4.3 Progressive Longest Common Subsequence

Metoda PLCSS [12] umožňuje stejně jako PDTW provádět zarovnání nad více než 2 sekvencemi najednou. Základní princip progresivní varianty metody LCSS je stejný jako u metody PDTW a tedy má stejné výhody a nevýhody.

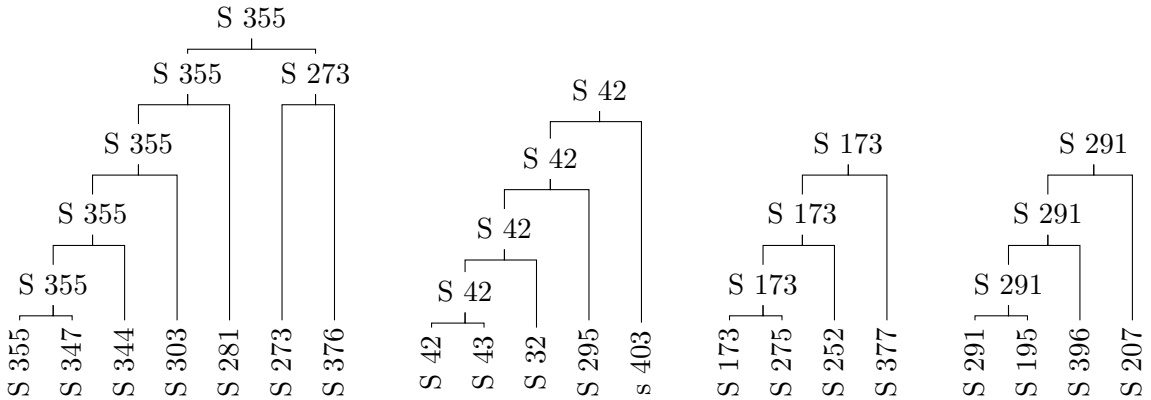


Obrázek 20: Znázornění progresivního zarovnání šesti sekvencí metodou PLCSS

Máme 6 sekvencí A až F, které chceme zarovnat, viz obrázek 20. Na počátku algoritmu je zkonstruována diagonální matice, která uchovává vzdálenosti mezi všemi páry zarovnávaných sekvencí. V diagonální matici je vyhledán nejbližší pár, který je zarovnán a sloučen do skupiny. Následuje aktualizace diagonální matice, pro zahrnutí informace o sloučení sekvencí, respektive skupin. Po té je opět vyhledán nejbližší pár, analyzován, sloučen atd.. Tento postup trvá, dokud nezbude jen jedna skupina zarovnaných sekvencí.

V našem příkladě jsou na začátku algoritmu nejbližší sekvence E a F, které jsou sloučeny do skupiny EF. Je zde vidět, že body, které jsou na sebe namapovány jsou rozdílné maximálně o 1, což je způsobeno parametrem $\varepsilon = 1$. V situacích, kdy jsou body příliš rozdílné jsou do sekvencí vloženy mezery (gaps) značené -. V případě progresivního zarovnání, takto vložené mezery v sekvencích nikdy nezmizí, propagují se až do výsledného zarovnání. V další iteraci jsou

nejbližší sekvence B a C, po které následuje sekvence D se skupinou EF. Stejně jako u metody PDTW je při slučování skupin, vyhledáno jedno reprezentativní zarovnání. Provede se tedy zarovnání nad sekvencemi DE a DF. Podle lepšího zarovnání se řídí zbylé sekvence v daných skupinách. Jak lze vidět, při vytváření skupiny DEF jsme získali sekvenci 87679--09-7-, vložením 3 mezer do sekvence 8767909-7, tedy mezery jsou vloženy i do sekvence 975--085-. V další iteraci jsou nejbližší skupiny BC a DEF. První sekvence ze skupiny BC a druhá sekvence ze skupiny DEF (nejbližší sekvence jsou označeny šipkami) vytváří nejlepší zarovnání. Mezery o které byly sekvence rozšířeny jsou opět vloženy do zbývajících sekvencí z daných skupin. Stejným způsobem pokračujeme, při vytváření zarovnání A a BCDEF.



(a) $Id = 545$ (b) $Id = 543$ (c) $Id = 492$ (d) $Id = 556$

Obrázek 21: Znázornění shlukování sekvencí pomocí metody PLCSS nad reálnými daty

Tabulka 7: Výsledky analýzy metody PLCSS nad shluky dat simulace prodejů UTP kabelů

Id shluku	Počet s.	Ratio	Id shluku	Počet s.	Ratio
545_l1	7	0,8750	577_l0	3	0,8750
543_l1	5	0,8750	562_l1	3	0,8666
492_l0	4	0,8214	768_l0	3	0,8461
556_l1	4	0,8269	571_l1	3	0,8666
588_l1	4	0,7857	571_l0	3	0,8933
552_l1	4	0,8750	600_l1	3	0,7976
580_l1	4	0,8076	609_l1	3	0,8518
578_l0	4	0,8148	566_l0	2	0,9411
596_l1	4	0,8703	628_l0	2	0,9473
586_l0	3	0,9607	603_l0	2	0,9230
608_l1	3	0,8518	599_l1	2	0,9411
610_l1	3	0,8666	510_l0	2	0,9130
527_l0	3	0,8666	525_l0	2	0,8750
572_l0	3	0,8750	584_l1	2	0,8518
588_l0	3	0,8333	532_l1	2	0,9231

Metoda PLCSS byla použita na analýzu reálných dat. Jedná se o nasbíraná data chování uživatelů v umělém ERP[3] systému. Konkrétně se jedná o data simulace uživatelů (umělá inteligence), kteří simulují prodej UTP kabelů. Každý uživatel je reprezentován sekvencí akcí, které v systému provedl. Z těchto dat byl vytvořen graf chování, který byl následně zredukován pomocí LCS a LCSS metod [11]. Výsledné shluky sekvencí byly analyzovány metodu PLCSS. Výsledky lze vidět na obrázku 21 a v tabulce 7.

[3]ERP (Enterprise Resource Planning) systém slouží k usnadnění řízení organizace. Konkrétně se jedná o podnikový informační systém, který řídí oblasti podniku jako je nákup, prodej, marketing, finance, skladování, výroba a doprava.

5 Implementace

V poslední kapitole této práce bude vysvětlena samotná implementace aplikace, tedy bude se zabývat strukturou aplikace a přístupem k samotné implementaci. Dále zde bude podrobně vysvětleno ovládání aplikace a požadavky na její úspěšný provoz.

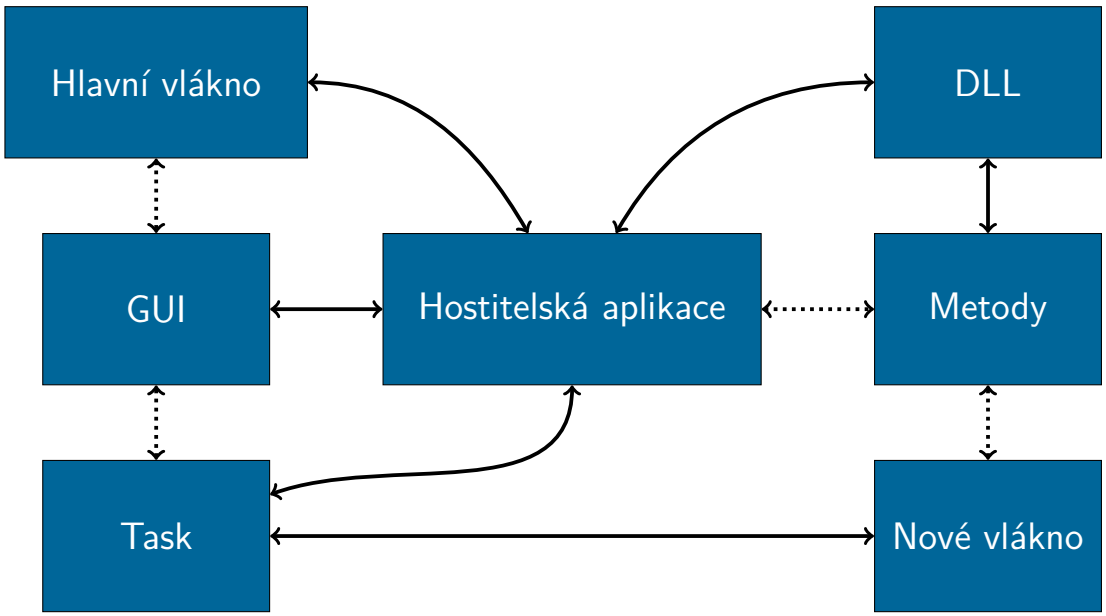
5.1 Programovací jazyk

Jako programovací jazyk jsem zvolil C# z platformy .NET. K samotné implementaci jsem použil .NET Framework 4.6 Client a jako vývojářské prostředí jsem použil Visual Studio 2015. Jednotlivé testy byly provedeny na počítači s core i5 3,4GHz a 16GB RAM.

- Podporované operační systémy pro .NET Framework 4.6 Client [16]:
  - Windows Vista SP2 (x86 a x64)
  - Windows 7 SP1 (x86 a x64)
  - Windows 8 (x86 a x64)
  - Windows 8.1 (x86 a x64)
  - Windows Server 2008 SP2 (x86 a x64)
  - Windows Server 2008 R2 SP1 (x64)
  - Windows Server 2012 (x64)
  - Windows Server 2012 R2 (x64)
  - Windows Server 2016 (x64)
  - Windows 10 (x86 a x64)
- Nepodporované operační systémy:
  - Windows XP a starší
- Minimální požadavky na systém:
  - Procesor s frekvencí 1 GHz nebo vyšší
  - 512 MB paměti RAM
  - 4,5 GB volného místa na pevném disku (x86)
  - 4,5 GB volného místa na pevném disku (x64)

5.2 Struktura aplikace

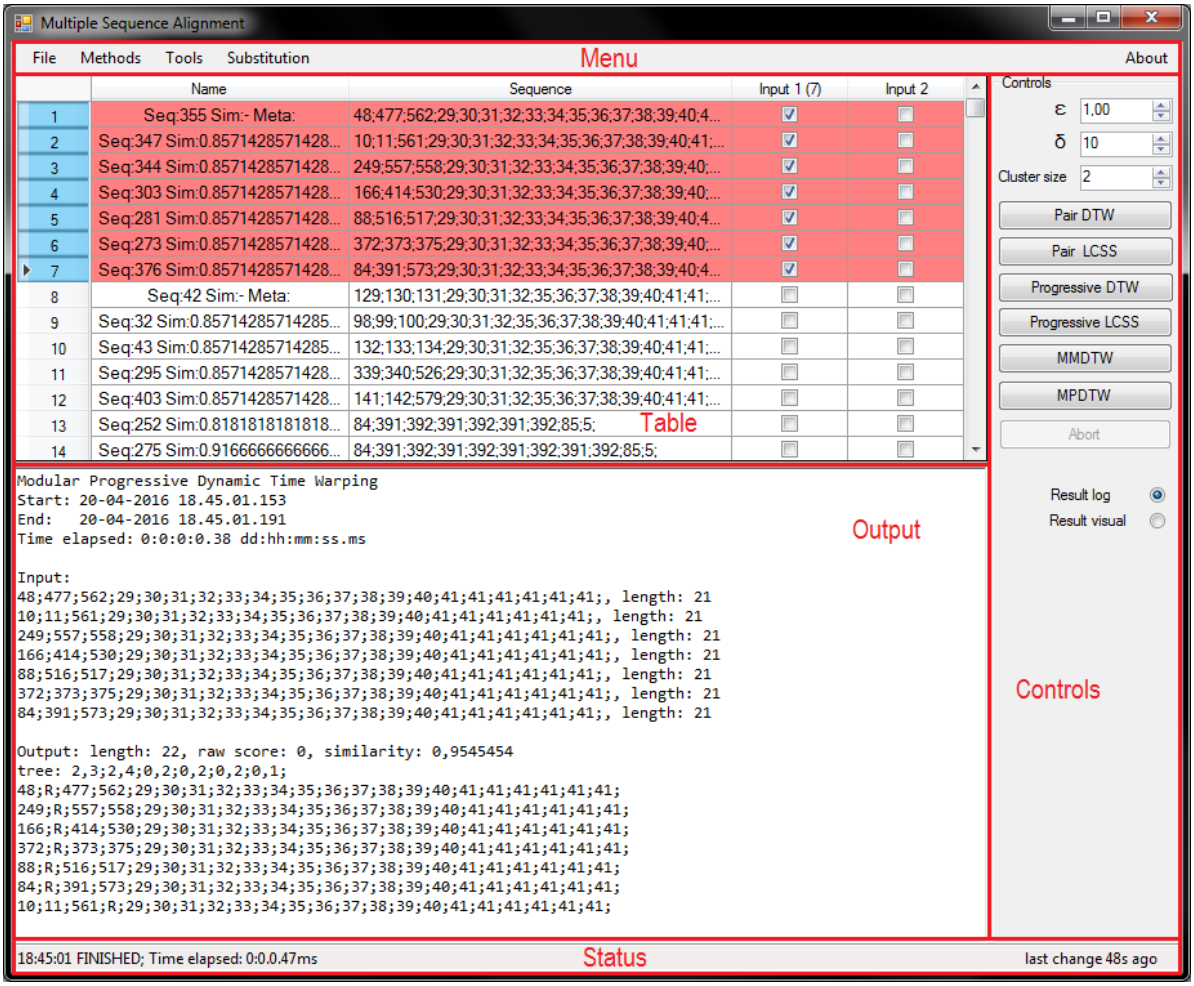
Implementace metod byla provedena ve formě DLL knihovny, z důvodu jednoduché přenositelnosti a možnosti importování metod do dalších aplikací.



Obrázek 22: Znázornění struktury programu

Jak lze vidět z diagramu 22, metody jsou z pouštěny prostřednictvím *Task* [14] pro zajištění, aby hlavní vlákno které má na starost GUI nebylo blokované běžícími metodami. Tasky jsou od .NET Framewokk 4.5 doporučený způsob, jak z pouštět metody ve vlastních vláknech. Na rozdíl od běžných vláken, Tasky jsou schopné přímo vracet návratovou hodnotu bez nutnosti vytvářet objekty.

5.3 Grafické uživatelské rozhraní



Obrázek 23: Náhled grafického uživatelského rozhraní

Grafické uživatelské rozhraní je složeno z několika částí:

- Menu - Nabídka s ovládacími prvky aplikace.
- Table - Tabulka pro manipulaci a výběr sekvencí určených pro analýzu.
- Controls - Ovládací prvky pro spouštění metod a nastavení parametrů pro dané metody. Také obsahuje ovládací prvky pro přepínání textových a grafický výsledků.
- Output - Pole pro zobrazení textových a grafických výstupů jednotlivých metod.
- Status - Status bar podávající informace o stavu aplikace.

Struktura a popis ovládacích prvků obsažených v menu:

- File - Základní operace s aplikací.
  - Open - Vývěr souboru se vstupními daty.
  - Reload input - Znovu načtení vstupních dat.
  - Add input - Načtení dalšího vstupu.
  - Save graph - Možnost uložit vizualizaci zarovnání metod DTW a LCSS.
  - Exit - Ukončení programu.
- Methods - Alternativní způsob spuštění metody pro analýzu sekvencí.
  - Dynamic Time Warping
  - Longest Common Subsequence
  - Progressive Dynamic Time Warping
  - Progressive Longest Common Subsequence
  - Multiple Multidimensional Dynamic Time Warping
  - Modular Progressive Dynamic Time Warping
- Tools - Nástroje pro usnadnění práce s aplikací.
  - Uncheck all - Zruší výběr všech označených sekvencí v tabulce.
  - Abort method - Přeruší vykonávání metody.
  - Always top - Aplikace zůstane v popředí i po ztrátě fokusu.
- Substitution - Nastavení jaká substituční matice je použita pro měření vzdálenosti mezi sekvencemi složených z kategoriálních dat.
  - Lexicographical
  - BLOSUM62
- About - Základní informace o aplikaci.

Ovládací prvky obsažené v části Controls:

- Parametry.
  - ε - Nastavení parametru epsilon pro metodu LCSS a PLCSS.
  - δ - Nastavení parametru delta pro metodu LCSS a PLCSS.
  - γ - Nastavení parametru gamma pro metodu MPDTW.

- Methods - Tlačítka pro spuštění jednotlivých metod.
  - Dynamic Time Warping
  - Longest Common Subsequence
  - Progressive Dynamic Time Warping
  - Progressive Longest Common Subsequence
  - Multiple Multidimensional Dynamic Time Warping
  - Modular Progressive Dynamic Time Warping
- Výběr mezi textovým a grafickým výstupem.
  - Result log - Textová část výstupu.
  - Result visual - Grafická část výstupu. Grafická vizualizace je dostupná jen pro metody DTW a LCSS. Navíc analyzované data musí být číselného typu.

5.4 Vstupní data

Při startu aplikace jsou automaticky načteny všechny datové soubory, které jsou zaznamenány v souboru *inputs.txt* ve stejné složce jako program $MSA.exe$. Jednotlivé datové soubory pak musí být umístěny ve složce *Data*, která je také umístěna ve složce s programem. Stejně jako v datových souborech symbol > značí komentář. Datové soubory musí splňovat [základní informace o sekvenci]@[sekvence] formát a tyto pravidla:

1. Každá sekvence je na separátním řádku.
2. Základní informace o sekvenci jsou uvedeny před symbolem @. Tyto informace jsou zobrazeny v tabulce ve sloupci $Name$.
3. Jednotlivé body v sekvencích jsou odděleny středníkem (;). Pokud je sekvence jedno dimenzionální a skládá se jen z jednociferných čísel, tak je možné středníky vynechat.
4. Jednotlivé dimenze v bodech jsou odděleny dvojtečkou (:). Všechny body v sekvenci musí mít stejný počet dimenzí.
5. Pro desetinná čísla je možné použít tečku nebo čárku (, nebo .).
6. Symbol > na začátku řádku slouží pro označení komentářů.
7. Komentáře a prázdné řádky jsou ignorovány.

Příklady formátování sekvencí v datových datových souborech:

- `random@CACCCGAAGGCGACAGTGCCTTATATACCCGTTGGA`

- `int 50@795533694328040967530156316226290387095335612806 32`
- `1.3.2015; 7; @117.43;117.43;117.43;117.43;117.43;107.92;101.77;`
- `num@2.0;3.0;7.0;7.0;8.0;8.0;6.0;2.0;5.0;2.0;4.0;5.0;5.0`
- `int d2@9:9;9:4;4:1;2:9;5:9;1:3;2:4;5:4;3:9;5:3;7:5;6:3;4:4;7:2;3:6;1:1;`
- `double d2@10.879848776317:2.109488263858;1.0959405201208:9.7551546135439;`
- `@131365496841331246421346`
- `d2@3,5:2,109;1,09:9,59;59:505;6226:575;`
- `test@-2,974;-2,1410;-9.0009;-7,8071;1.8930;-1.75;`

Případ spuštění metody:

1. Vybrání sekvencí pro analýzu.

    - Minimální počet vybraných sekvencí pro analýzu jakoukoli metodou je 2. Pokud používáme metodu DTW a LCSS, tak se berou první dvě vybrané. Pro ostatní metody se berou všechny označené. Označení se provádí přes sloupec *Input* 1.
    - *Input* 2 slouží jen pro případ, kdy uživatel chce přehodit pořadí analyzovaných sekvencí pro metody DTW a LCSS. Sekvence jsou tímto přehozeny i v grafickém výstupu.
    - Analyzované sekvence musí být všechny buď číselné nebo kategoriální. Pokud jsou analyzované sekvence různých typů, tak běh metody skončí výjimkou.
    - Pokud mají analyzované sekvence různý počet dimenzí, tak se nadbývající dimenze ignorují.

2. Nastavení parametrů pro naše potřeby v sekci *Controls*.

3. Spuštění metod je možné buď přes tlačítka v sekci *Controls* nebo přes *Menu* pod nabídkou *Methods*.

4. Po úspěšném výpočtu metody jsou výsledky zobrazeny v sekci *Output* a zároveň jsou zapsány do souboru *log.txt*. Případná možnost přerušení výpočtu právě běžící metody pomocí tlačítka *Abort*.

6 Závěr

Cílem této práce bylo seznámení se s problematikou zarovnávání sekvencí, jak ze strany teoretické, tak jako praktickou zkušenost při následné implementaci několika vybraných metod. Metody, kterými jsem se zabýval v této práci jsou Dynamic Time Warping a Longest Common Subsequence spolu s jejich vícenásobnými variantami. Samotné metody jsem implementoval jako DLL knihovnu, která následně byla využita hostitelskou aplikací pro práci s danými implementovanými metodami. Nad jednotlivými metodami jsem provedl množství testů, jak na náhodně generovaných datech, tak nad daty z reálného provozu. Metodu PDTW jsem testoval na zaznamenaných rychlostech provozu na vybraných místech v Anglii. PLCSS metoda pak byla testována na datech chování uživatelů v ERP systému. Výsledky těchto testů jsou shrnuty v podkapitolách 3.9 a 4.3.

Tabulka 8: Přibližná paměťová a časová složitost DTW a LCSS metod

Metoda	Typ	Časová složitost	Paměťová složitost
DTW	párová	$O(m^2)$	$M(m^2)$
LCSS	párová	$O(m^2)$	$M(m^2)$
PDWT	progresivní	$O((n^2+n-2)m^2)$	$M(m^2)$
PLCSS	progresivní	$O((n^2+n-2)m^2)$	$M(m^2)$
MMDTW	vícenásobná	$O(m^n)$	$M(m^n)$
MPDTW	vícenásobná progresivní	$O((\frac{n^2+n}{2})m^2+(\sum_{i=1}^{\frac{n}{\gamma-1}}(\gamma^i)^{k-1})m^\gamma)$	$M(m^\gamma)$

Kde m je délka sekvencí, n je počet sekvencí a γ je parametr počtu shlukovaných sekvencí.

V tabulce 8 lze vidět srovnání přibližné časové a paměťové složitosti jednotlivých implementovaných metod. Jak lze vidět paměťová složitost PDTW a PLCSS metod zůstává na úrovni párových metod. MPDTW metoda má sice stále paměťovou složitost exponenciální, ale jelikož exponent k je škálovatelný, tak se v praxi dostaneme na složitosti n^3, n^4, atd., které jsou relativně zvládnutelné. Většina progresivních metod platí za menší časovou složitost tím, že mají nižší přesnost výsledků než metoda MMDTW. Je tedy potřeba zvážit, jaká metoda je pro konkrétní data nejvhodnější.

6.1 Možnosti dalšího vývoje

Tato práce se zaměřila na studium a implementaci metod, které využívají dynamické programování. Bohužel dynamické programování je ve formě, v které je využívají metody vysvětlené v této práci velmi nevhodné pro paralelizaci. Hlavní důvody jsou nutnost uchovávání velkého množství dat v paměti a problematické rozložení výpočtu matice vzdáleností na více vláken,

z důvodu nutnosti čekat na předchozí výpočty. Další vývoj by se tedy mohl zaměřit na paralelní implementaci, která by s největší pravděpodobností spočívala v eliminaci dynamického programování.

S ohledem na to, že jsem se zaměřil jen na globální zarovnání, tak by následující vývoj určitě také spočíval v přidání podpory pro lokální zarovnání, které je v mnoha případech žádanější než globální.

Další možnost vývoje je větší generalizace analýzy. Implementovaná aplikace momentálně umí pracovat jen s číselnými a kategoriálními sekvencemi. Tedy umožnit práci se sekvencemi, které například mají body složené z více dimenzí a každá dimenze obsahuje různé typy dat. To by si následně vyžádalo přidání podpory pro definování vlastních metod pro určování vzdáleností mezi body těchto generalizovaných sekvencí.

Touto prací jsem se dotknul jen velmi malé části oblasti zarovnávání sekvencí a mohl bych tedy strávit velké množství času při dalším studiu.

Martin Rusek

Literatura

[1] SANGUANSAT, P. *Generalized Dynamic Time Warping for Multiple Multidimensional Sequence Alignment*. In: *3rd internationalconference on Machine Learning and Computing*. Thailand: Nonthaburi, 2011, s. 426 - 431.

[2] SANGUANSAT, P. *Multiple Multidimensional Sequence Alignment Using Generalized Dynamic Time Warping*. In *Transaction on Mathematics (WSEAS)*. Thailand: Nonthaburi, 2012, s. 668 - 677.

[3] PETITJEAN, F., a Pierre GANÇARSKI. *Summarizing a Set of Time Series by Averaging: from Steiner Sequence to Compact Multiple Alignment*. France: Strasbourg, 2011.

[4] MÜLLER, Meinard. *Information retrieval for music and motion*. Berlin: Springer, 2007. ISBN 978-3-540-74047-6.

[5] VLACHOS, Michail., M. HADJIELEFTHERIOU, D. GUNOPULOS a E. KEOGH. *Indexing Multi-Dimensional Time-Series with Support for Multiple Distance Measure*. California: Riverside, University of California.

[6] WANG, Hiu. *All Common Subsequences*. In: *Proc. of the 7th International Joint Conference on Artificial Intelligenc (IJCAI-07)*. India: Hyderabad.

[7] GUNOPULOS, D. a D. GAUTAM. *Time Series Similarity Measures - Tutorial PM-2*.

[8] VLACHOS, Michail., G. KOLLIOS a D. GUNOPULOS. *Discovering Similar Multidimensional Trajectories*.

[9] JURAFSKY, Dan. *Minimum Edit Distance*.

[10] HENIKOFF, S. a J. G. HENIKOFF. *Amino Acid Substitution Matrices from Protein Blocks*. PNAS 89, s. 10915 – 10919.

[11] KATEŘINA, S., J. MARTINOVIČ a M. GOLASOWSKI. *Reduction of User Profiles for Behavioral Graphs*.

[12] DEWEY, Colin. *Multiple Sequence Alignment*. 2011. Dostupné z `https://drive.google.com/open?id=0B_gbaS2iMV7hRU52V2lkN2xjYnM`.

[13] XIE, Jun. *Pairwise Alignment*. Dostupné z `https://drive.google.com/open?id=0B_gbaS2iMV7hLVZJb0ZURUxQd3M`.

[14] *Task Class*. (n.d.). Dostupné z `http://msdn.microsoft.com/en-us/library/system.threading.tasks.task(v=vs.110).aspx`.

[15] *Phylogenetic Trees*. (n.d.). Dostupné z https://drive.google.com/file/d/0B_gbaS2iMV7hOU5FS1VDNUJwQTg/view.

[16] *.NET Framework System Requirements*. (n.d.). Dostupné z http://msdn.microsoft.com/en-us/library/8z6watww(v=vs.110).aspx.

[17] *Highways Agency network journey time and traffic flow data*. (n.d.). Dostupné z https://data.gov.uk/dataset/dft-eng-srn-routes-journey-times.

7 Seznam příloh

- Obsah CD
- Ukázka dat pro metodu DTW
- Ukázka dat pro metodu LCSS

8 Obsah CD

- Dokumentace - PDF obsahující textovou část diplomové práce.
- Program - Obsahuje složky zdroj, exe, testy.
  - Program/zdroj - Obsahuje zdrojové kódy programu.
  - Program/exe - Obsahuje spustitelný program.
  - Program/testy - Obsahuje soubory s testovacími daty pro program.
- DTW - Obsahuje data pro metodu DTW.
- LCSS - Obsahuje data pro metodu LCSS.

9 Ukázka dat pro metodu DTW

Zaznamenané rychlosti provozu na vybraném místě ze dne 1. 3. 2015:

117.43;117.43;117.43;117.43;117.43;107.92;101.77;102.66;98.88;99.44;94.01;111.87;
111.60;107.60;113.88;97.32;98.28;99.41;103.90;106.90;104.15;107.72;112.73;106.41;
107.50;105.16;111.03;84.11;84.11;84.11;84.11;97.54;116.07;116.07;116.07;116.07;
106.47;104.06;104.06;104.06;104.06;104.59;105.13;105.13;110.56;110.56;110.56;
114.25;104.82;104.82;104.82;113.87;124.64;118.99;99.77;111.00;132.15;132.15;132.15;
116.05;109.93;126.69;126.69;126.69;119.56;113.20;112.56;111.94;111.94;90.11;90.11;
90.11;90.11;94.28;94.28;94.28;101.15;109.09;109.09;113.60;118.50;118.50;118.50;118.50;
99.95;112.77;114.23;110.89;106.76;100.77;112.97;113.55;108.57;103.26;110.10;108.89

Poznámka 5 Všechna data pro metodou DTW jsou dostupná na přiloženém CD.

10 Ukázka dat pro metodu LCSS

Id: 545_l1 Elements: 7
Seq:355 Meta:48;477;562;29;30;31;32;33; 34;35;36;37;38;39;40;41;41;41;41;41;41;
Seq:347 Meta:10;11;561;29;30;31;32;33;34;35;36;37;38;39;40;41;41;41;41;41;41;
Seq:344 Meta:249;557;558;29;30;31;32;33;34;35;36;37;38;39;40;41;41;41;41;41;41;
Seq:303 Meta:166;414;530;29;30;31;32;33;34;35;36;37;38;39;40;41;41;41;41;41;41;
Seq:281 Meta:88;516;517;29;30;31;32;33;34;35;36;37;38;39;40;41;41;41;41;41;41;
Seq:273 Meta:372;373;375;29;30;31;32;33;34;35;36;37;38;39;40;41;41;41;41;41;41;
Seq:376 Meta:84;391;573;29;30;31;32;33;34;35;36;37;38;39;40;41;41;41;41;41;41;

Id: 543_l1 Elements: 5
Seq:42 Meta:129;130;131;29;30;31;32;35;36;37;38;39;40;41;41;41;41;41;41;33;34;
Seq:32 Meta:98;99;100;29;30;31;32;35;36;37;38;39;40;41;41;41;41;41;41;33;34;
Seq:43 Meta:132;133;134;29;30;31;32;35;36;37;38;39;40;41;41;41;41;41;41;33;34;
Seq:295 Meta:339;340;526;29;30;31;32;35;36;37;38;39;40;41;41;41;41;41;41;33;34;
Seq:403 Meta:141;142;579;29;30;31;32;35;36;37;38;39;40;41;41;41;41;41;41;33;34;

Id: 492_l0 Elements: 4
Seq:252 Meta:84;391;392;391;392;391;392;85;5;
Seq:275 Meta:84;391;392;391;392;391;392;391;392;85;5;
Seq:173 Meta:84;391;392;391;392;391;392;391;392;391;85;393;5;
Seq:377 Meta:84;391;392;391;392;391;392;391;392;391;392;85;5;

Id: 556_l1 Elements: 4
Seq:207 Meta:324;450;451;450;452;29;30;31;32;35;36;37;38;39;40;41;41;41;33;34;41;41;41
Seq:291 Meta:137;138;413;29;30;31;32;35;36;37;38;39;40;41;41;41;33;34;41;41;41;
Seq:195 Meta:42;435;436;29;30;31;32;35;36;37;38;39;40;41;41;41;33;34;41;41;41;
Seq:396 Meta:289;290;577;29;30;31;32;35;36;37;38;39;40;41;41;41;33;34;41;41;41;

Poznámka 6 Všechna data pro metodou LCSS jsou dostupná na přiloženém CD.